[Comment #1] Software scope

There is a mention of software scope on lines:
25, 28, 75, and specifically:
167- 170

> *Note that while the concepts and relationships described in this reference model may apply to other "service" environments, the definitions and descriptions contained herein focus on the field of software architecture and make no attempt to completely account for use outside of the software domain. Examples included in this document that are taken from other domains are used strictly for illustrative purposes.*

These 4 references [ed: 25,28,75, 167-170] are not enough to clarify this scope to the reader. Specifically, the use of terms such as capability and need which are generic to any domain and the accompanying examples that are from everyday life and not necessarily from software domain, re-enforce to the reader the false impression that this reference model and/or SOA principles in general apply to any domain where a service might be used. I find this adds to the current misconceptions within DoD that view SOA as the answer to all problems, not realizing the principles and techniques set forth in industry and by this body (OASIS) are restricted to the software domain.

Suggestions:
1. Make the scope clearer by adding stronger language to introduction and to section 2.1
2. remove examples from every day life and make them clearly examples for software solutions
3. change the use of the terms need and capability to change *need* to business process and *capability* to service or function

[text at end supplied by commenter in separate email]

Lines 58-60, scope being software only can be reinforced here.
Line 154: again implies services could be those provided by humans for example (confusing)
Line 272: “Thus, the service could carry out its described functionality through one or more automated and/or manual processes that themselves could invoke other available services.”
This is clearly Not a software service if a manual process is involved to perform the service function. The use of a service may involve a human, but the service itself is not a manual process. This again adds to the confusion about

the definition of a service.
Line 277: real world effect again implies scope outside software. Usually effect on real world is indirect.

PROPOSED RESPONSE:
DN: We have not restricted it [the RM] to software. We state that it may be valid/applicable outside but we focus only on software and do not fully explore the other applications.
DN: In general, it cannot be labeled that it is a false impression that the RM for SOA might be applicable to domains outside software where a service is used. In fact, I find that they are often highly relevant. If I am going to open a coffee shop, the model is very applicable. A service (providing coffee to customers) still needs to be made visible (service description) and has an interaction model (customer states order and pays cash, coffee shop returns a caffeinated beverage) and other aspects of the RM for SOA. We do explicitly state that

> *"The concepts and relationships described may apply to other "service" environments; however, this specification makes no attempt to completely account for use outside of the software domain."*

KJL: With regard to "the current misconceptions within DoD that view SOA as the answer to all problems", the RM includes lines 160-165 to explicitly emphasize the counterpoint.

> *SOA is a means of organizing solutions that promotes reuse, growth and interoperability. It is not itself a solution to domain problems but rather an organizing and delivery paradigm that enables one to get more value from use both of capabilities which are locally "owned" and those under the control of others. It also enables one to express solutions in a way that makes it easier to modify or evolve the identified solution or to try alternate solutions. SOA does not provide any domain elements of a solution that do not exist without SOA.*

With regards to questions about the human or manual role in services:
DN: They [humans] can [be provided by humans]. Managed transparency is a core aspect of SOA – you do not know how the service is provided, only that it is. The service provider decides how much you get to know and hides the rest. The lines between humans and software are blurry. It is possible that a services back end has a human being flagged to review and approve a travel permission form submitted by an employee (I actually just did this). The person submitting this should not really care, only that they know they have to use the service.
DN: Here we must disagree. The concept of opacity is that the consumer of a service has no idea how the actual service gets carried out and it can be done manually via a software interface.

KJL: +1 The commenter comes from a DoD background where they have arbitrarily (and I have argued, incorrectly) limited their attention to what they assume to be machine-to-machine, with the human interaction restricted to a portal. The idea has not taken hold , for example, that error recovery could be routed to a human today and later be done by a machine without any explicit signs of this change.

With regards to real world effect:
DN: The TC feels very strongly that the real world effect of a service interaction is a core aspect of SOA. Even in software architecture, the software's use typically results in some real world effect (a record being created, a business process advancing one step, a user account being created etc.) The concept's are inseparable, although in an abstract reference model, we only note the abstract concept of the RWE, not specific real world effects.

PROPOSED DISPOSITION:
No change to text and send response back to commenter

Comment # 2:
Use of the term architecture within document to denote an arch model (see lines 34-37). A blue print is a (partial) model of the house. One blue print only shows one perspective, and several of those actually model the complete architecture. The house itself has a certain architecture (the concept), but that architecture is not the blueprint itself. Pls see IEEE 1471. Restrict use of term architecture to mean the actual concept of the thing being described. And/or clarify at beginning that use within this document is always to architecture model, but that word the "architecture" is being used for brevity to actually mean architecture model.
· IEEE1471: "Things" (called systems in the standard), "Architectures," and "Architecture Descriptions" (AD)
— Things are systems, enterprises, systems-of-systems, man-machine systems, whatever you are interested in - "System of interest".
— "Architecture" is a conceptual property of a Thing.
· Tied to the structure of the Thing, but not necessarily the physical structure.
· A Thing (read a system) has one architecture
— An "Architecture Description" is a document/model of the Architecture of a Thing

DN: Since we used architecture in a lexically scoped manner, we included a formal definition in our glossary as "A set of artifacts (that is: principles, guidelines, policies, models, standards and processes) and the relationships between these artifacts, that guide the selection, creation, and implementation of solutions aligned with business goals. Software architecture is the structure or structures of an information system consisting of
entities and their externally visible properties, and the relationships among them."

We are happy with that definition since it was used consistently throughout the specification. We would not benefit from changing or aligning with the IEE 1471

lexicon although we recognize that there are other definitions. IN general this was one of the most highly discussed definitions in the glossary.

PROPOSED: NO ACTION, reply to commenter

KJL: wouldn't want to cut this off too abruptly. If aligning more closely with 1471 is easy and, in particular, if it lends some clarity, I think we should consider rewording our text, although I wouldn't reference 1471 directly. I asked someone else to consider this.

PROPOSED INTERIM ACTION:
consider what it would mean to align more closely with 1471

Comment #3 (related to 1 above)
[Given the following excerpts from the RM]:
Line 262 definition of term service: "A service is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description."
Line 530: "That the service performs a certain function or set of functions;"
Line 543: "3.3.1.2 Service Functionality"
Line 550: functions

[commenter notes and suggests]
1. There seem to be two different definitions of the term service-capability and function.
2. Suggest you stay away from using the term capability and use term function instead (more closely associated with software than is the term capability).
On Line 474: internal actions (use implementation)
Line 510: you use term function to mean HOW the service is provided which is internal to service implementation (same as your term internal action on line 474), but should not be confused with term "function" used on line 530 to mean the service that is actually provided. I would assert that function DOES NOT change.
Line 518, use term function as opposed to capability (for consistency in document), especially sine you end sentence with term functionality

DN: They are two different things. A capability is precisely that. The function represents the abstract concept of the mechanism that uses the capability. We made a change to your previous comment however which will clarify the issue a lot. There was some ambiguity which that will solve. Capability is defined in the glossary on 741.

KJL: Hope this doesn't open a can of worms. Function is what it does and leads to RWEs; capability is the magic that does it. A service access a capability in order to carry out a function.

KJL: [re using "implementation" in Line 474] While implementation is an aspect, the discussion is more encompassing to any actions, whether through a single implementation or a series of implementation used together.

KJL: [re use of "function" in Line 510] Function was used loosely here to refer to something that could be described through the service description. It is not meant to be HOW the service is provided and is meant to be more in line with my comment regarding Line 550. Both are indeed in the same context as line 530. Finally, the text is not meant to mean the service functionality would change but rather service descriptions of different services could include more or less detail on any the elements we connect with service description.

DN: [re use in Line 518] Disagree with comments on change to line 518 from "Capabilities" to "functions". It is not the exact same thing.

PROPOSED DISPOSITION:
No change to text and send response back to commenter

[additional comments on cardinality]
DN: Commenter is correct to point this out. PROPOSED CHANGE: remove reference to (or set of) functions at line 530 as it implies cardinality in order to be consistent.

KJL: -1 this is just an excerpt and the issue is expressed later. WRT cardinality here, I think it is necessary to emphasize that there is not necessarily a one-to-one correlation. Similar emphasis appears elsewhere in the text for other things.

IMMEDIATE ACTION: need to resolve

Comment #4
Lines 262 and 530: Contend that a service should provide ONLY one function but multiple interfaces (line 268 implies only one interface), otherwise it is not properly defined and well encapsulated. A function does not necessarily mean one at the level of a C++ method for example.

DN: I am at odds on whether or not this is concrete given it has cardinality constraints. Perhaps we need to change this to only illustrate that there is a relationship between the concepts of interfaces and functions, regardless of cardinality.

Line 262 reads:
"A **service** is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description."

KJL: A given interaction with a service is through one interface although other interaction may use other interfaces as appropriate. As far as a service providing only one function, it is up to the service provider to decide what is a relevant function or possible set of functions that a consumer may recognize and want.

DN: Change line 262 to
"A **service** is a mechanism to enable access to (…deleted…) capabilities,…"

KJL: -1 As noted in a comment above, the intent of the wording is to specifically say there is no cardinality constraint.

IMMEDIATE ACTION: need to resolve

Comment #5
Line 264: "A service is provided by one entity – the service provider –." More than one provider could provide same service. Perhaps you mean to just define the term provider. But your words imply there can only be one.

DN: commenter is correct. Once again there is a cardinality aspect to this and we need to remove it given this should be abstract.

PROPOSED CHANGE:

Line 264: "A service is provided by an entity called the service provider –."

Given the abstractness of the model, it will be inferred that this could be 1-* relationship in a more concrete spec based on the RM.

KJL: No, "an" implies a cardinality of one. Also, assuming the point of interaction is a specific endpoint, "a" service is provided by one and only one service provider. If the endpoint changes, something about the service changes even if the provider remains the same. Now, the same functionality may be provided by many competing services, some of which may access the same underlying capability.

Considering the wording, the article or adjective and whether singular or plural implies a cardinality unless you specifically say one or more.

IMMEDIATE ACTION: need to resolve

Comment #6
Line 281: the use of the term change in shared state is confusing and may mean state of service itself. The service does not change state as a result of being invoked and then executing. If it did, others will not be able to use it

afterwards. Suggest: state of object of service is changed.
Line 364: related comment: assumes service provider has authority to effect change in state of some object that consumer needs, this authority is given by whom? It is more likely that provider provides some computational service, the results of which are passed back to consumer who then stores it in some object (which he owns, or has authority over) to effect a change in state of that object. My point is to avoid use f term shared state. If services are truly agnostic of consumers, they will have no knowledge nor authority to effect any changes directly, and will not directly effect a state change.
Line 452- editorial: remove extra and "…between and temporal properties…"

DN: Ken is taking care of this with his comments. Noted that this is confusing. We will fix.
KJL: As noted in resolving Issue 539, the TC considered alternatives to the term "shared state" and found none that would not introduce other concerns. The term will not be changed but text has been added to clarify the concept as it applies to the RM.

As to specific comments:

KJL: [re line 281] The consequence may be a change in shared state and I would not rule out (nor would I advocate) a situation where the interaction actually did change the state of the service. For example, I contract to use a service to download 50 songs. After the 50th download, the state of the service may change to block further downloads.

KJL: [re Line 364] This is implementation and business rule specific implying a best practice. At the RM level, there is no reason to include such a restriction.

KJL: [re line 452] intent was to specifically call out relationships AND properties. I seem to remember a comment on this before and we either decided not to change or this reflects the change. Frank?