ears and Eventual Acceptance of CMMI

Paul D. Nugent, Ph.D.

General Dynamics Advanced Information Systems,

Rensselaer Polytechnic Institute

ears and Eventual Acceptance of CMMI

- Research Question
- Setting/Method
- Analysis
- Conclusions

ears and Eventual Acceptance of CMMI

- Original Research Question
  - Organization Theory
    - Formal organization – *codified* process, org. chart, etc.
    - Informal organization – *unwritten*, cultural, normative
      - Responsive to importance and uncertainty
    - Systems of control
      - Direction
      - Evaluation
      - Correction
    - CMMI as formalization of process previously controlled by informal organization
- Initial reactions
- Follow-up 3 Years Later

ears and Eventual Acceptance of CMMI

- Setting/Method
  - Setting
    - Defense contracting, complex embedded submarine systems, New England
    - ~1000 Employees
    - >70% professional (engineers/managers)
  - Method
    - Ethnographic, grounded theory, participant observation

ears and Eventual Acceptance of CMMI

- Analysis
  - Specific Phase I patterns
    - Production of artifacts
    - Assumption of a perfect product
    - Performance evaluation and protection
    - Promotion of self-interest
    - Vertical migration of task definition
  - Phase II
    - Strength of informal organization (culture/structure)
    - Strong norms regarding customer expectations and product quality
    - Intelligent integration of CMMI with other processes (peer review databases, development tools [DOORS])

ears and Eventual Acceptance of CMMI

- Conclusions
  - Evaluative facet of CMMI process initially threatens to erode community and hinder productivity
  - If informal organization is effective and strong, these negative effects are avoided and CMMI’s performance enhancing features, as a tool, are embraced

ears and Eventual Acceptance of CMMI

Production of Artifacts

PHASE I

"It did not change how we develop software. It changed the artifacts that are produced, it changes individual *accountability* for producing *artifacts*. It drives up the costs in terms of the number of hours you have to expend- that cost has to be absorbed by the individual developers because costs can only go down – never up. The rationale is that this is going to make us *cheaper* and to do that – and they really just put more pressure on individuals to follow more religion, produce more artifacts – so what it comes down to is software *always* developed the same way and you can tell that by watching a project. "

PHASE II

felt while some of the "overhead" is undesirable and takes away from the "real work," by and large this "tax" was not a large burden. Furthermore they felt that it did not sap the effectiveness of the informal process. Engineers still consulted with one another and performed informal reviews as they had done in the past. Incorporation into DOORS and peer review tools minimized overhead.

BACK

Assumption of a Perfect Product

PHASE I

"So you go into a room with a work product, and *anything* that you say is wrong with it is considered a defect. That means that if somebody feel that if it's appropriate, there'll be a root-cause analysis. "Why did you do this?" So you *replace* the useful engineering discussion – a work group with something that is a lot uglier. You know, it's based on the assumption that there shouldn't have been anything wrong in there. The point of the process is to *improve* the product. There's the expectation now that the product will go in there near perfect."

PHASE II

As CMMI processes became institutionalized and accommodated by the informal processes, it became clear that informal pre-reviews largely mitigated these negative effects by being positive, non-critical, and constructive.

BACK

Performance Evaluation and Protection

PHASE I

"Although everybody's assured this doesn't happen, and it's denied vehemently, people know that management goes into these databases and looks for *individual* producers and defects as part of how to establish who their high performers are – who produces a quality product. That makes people very defensive when they go into these forums."

PHASE II

"I try to call everything minor. Because if you flag something as a major, if somebody's looking at that database that can be more of a mark against an individual. Again, from a Systems point of view – what discriminates the two, there's a definition in the process as to what the severity is, but ultimately it's a subjective call. You try to justify something being a minor however you can. Nobody likes to say there's a major defect in somebody's work product."

BACK

Promotion of Self-Interest

PHASE I

"If you have a milestone to meet. If you have an artifact to produce and regardless of where your true commitment falls, when somebody draws a line in the sand and says you have to get something done by a particular point in time and then there's something else that *really* should be done by a particular point in time but doesn't directly reflect on that artifact, where do you place the priority? You prioritize based on what you're being measured to when push comes to shove. If what you really needed to do was take the time to work with somebody else to work out a critical interface, … let me back up. It promotes more self-interest. If individuals are being measured with milestones rather than measuring the *crew*, you know, based on what they really accomplish, people are going to *act* individually. It forces that reaction. You can't survive in that type of environment if you don't accomplish what you're individually accountable for. So everybody to some extent strives for a balance between the two. There are some people that try to be completely rebellious to that and those people end up well – they're *outside* of the organization after a short period of time."

PHASE II

The phase II data showed no increase in self-interested behavior, although the data gathered were not quantitative. The author attributes this to the more basic finding that the informal community/organization at the heart of the systems engineering tasks maintained its culture and structure and therefore norms dictating self-interested versus altruistic behavior were not changed.

BACK

Vertical Migration of Task Definition

PHASE I

“The other observation that I think is worth looking at is that this could have a detrimental impact on the capabilities of this organization. And maybe this is what separates the notion of the hero from somebody else. But the more they’re building a mentality that developing a product is turning a crank – executing to the process – the more individuals are going to rely on being spoonfed. The design of a system is really nothing more than working with different levels of abstraction at different points in time. For example, when you *do* lay something out at a high level you *rely* on individual software engineers – the people responsible for something, to solve significant portions of the their own problem – to finish coloring in the details on their interface. The more people expect something to be a matter of turning the crank, the more they look to other people to have something defined for them. So that taps the resources to go down to an unreasonable level of detail. To go down multiple layers of abstraction, beyond where they really should be to solve other people’s problems because the organization’s becoming less capable.”

PHASE II

Since, from the point of view of the customer, the “deliverables” did not need to carry with them CMMI artifacts, a strong pattern emerged in which the technical content of the product was the highest priority and if time ran out, the formal peer reviews and artifact gathering was postponed until after the official delivery.

BACK