International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

Carrier Grade Firmware

Shashank Vimal
[1,2]*Senior Manager, Firmware Development, SSIR, Bangalore, Karnataka*

Date of Submission: 05-12-2020 Date of Acceptance: 20-12-2020

ABSTRACT: In this paper, definition, need, characteristics, significance, taxonomy and dependencies of carrier grade firmware, will be discussed. Relationship of carrier grade firmware with extreme embedded system and deeply embedded system, will be established. This paper will also highlight methodologies for assessing whether a firmware component can be claimed to be carrier grade firmware (CGF)or not.
Firmware engineering is an indispensable and niche realm of semiconductor industry. There is similarity and disjoint between software and firmware engineering. Thus, while the software/IT industry adopts practices from its sister industry; semiconductor industry may not adopt practices and methodologies and apply them in their original form to its functional areas. So there exists an obvious disjoint between carrier grade software and carrier grade firmware. We will discuss that too, here.
The paper starts with discussion on reliability, availability and scalability(RAS) and then relates it to CGF. The levels of areas of assessments (AoA) will all be defined using first order logic and shall be self-explanatory.
KEYWORDS: Carrier grade,Extreme system**,** Deep Embedded System,ShadowDevice Driver, Chicken bit,Hardware accelerator, SCR latch-up, OAM&P.

I. INTRODUCTION

Firmware is an intangible component of modern processing element(PE), based systems which is comprised of a PE interpretable set of procedures and algorithms. This set of procedures and algorithms perform dedicated task and mostly remains unchanged or changes in a controlled environment with a very low frequency of change. That is how firmness is guaranteed into firmware. Embedded and/or real time systems and firmware goes hand in hand.

As we know that failure of realtime systems can have catastrophic effects, thus, firmware on-boarded on to a real time or embedded system has the same impact.

Like other engineering streams, firmware engineering also relies on various parameter measurements and the design decision depends on the inter-relationships of variables associated with the underlying system which houses the firmware, for e.g., current consumption, voltage, memory, time and also the ambient factors of the system hosting the firmware, such as temperature, vibration, humidity, pressure to name some.Firmware has a constrained and finite resources to satisfy the properties of the desired procedures that it comprises of. Whereas, all of this is not true for software.

Quintessentially, embedded systems house firmware and embedded systems has responsibility of performing a dedicated task for the users and owners of these systems. The embedded term comes from the fact that these systems have a processing element embedded into it. There exists a wide gamut of embedded system. Broadly, it can be classified as large-scale, mid-scale and small-scale system and based on QoS, further as, extreme embedded system and deep embedded system.

Extreme embedded systems are reactive, stateful, embedded and distributed. They requirereliability, availability and scalability(RAS) as a mandatory QoS. Cellular systems, Short range wireless systems, wireless sensor networks etc.are examples of extreme embedded systems whereas Deep embedded systems are tailored with application specific IPs and need low power consumption, performance and small memory footprint and has RAS as important concern for e.g., SIM card and micro-SD cards. The firmware which offers RAS to extreme and deep embedded system at any scale is said to be carrier-grade.
[1]The reliability of a system is basically the quality retention property of the system and is evaluated against time.
More elaborative definition of reliability is: [2]Reliability is a measurement of the probability that a system operateswithout failure over a specified time, within a specified environment, for a specified purpose. Reliability can be measured in terms of Mean time between failure(MTBF),

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

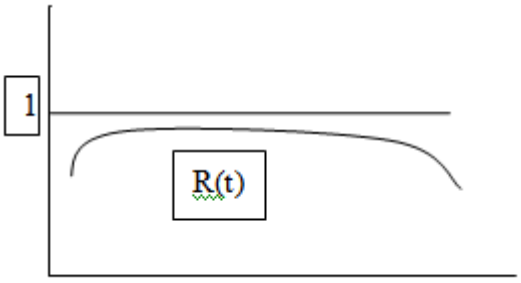
Probability of failure on demand(POFOD) or Rate of occurrence of failure(ROCOF).

Another reliability definition [3] the probability that a component does not fail until sometime t, is called reliability of the component, represented by R(t).

[6] For a system comprising many components, the reliability will be product of reliabilities of the individual components.

Being a dynamic parameter, reliability would keep on changing depending on failure rate of sub components, the cumulative distribution of failures over the set of sub components and the atomic fixes(solutions free from domino's effect) that is provided in the form of firmware patch.

Depending on the carrier grade maturity level and a myriad range of other factors such as maturity level of organization (producing such systems or products), the reliability will follow different probability distributions and plots. Typically, from Poisson, Binomial to an inverted bath tub curve.



If the solutions provided to firmware improves the reliability, without any snowball effect or demand for new firmware fixes, then the rate of occurrence of failure would descend leading to reliability improvement. But by and large, embedded systems have a fixed lifetime as the underlying hardware parameters changes due to aging, stress and the frequency of operation, on unusual operating points, violating the specified norms and that's the reason we see a steep obtuse slope in reliability towards the end of the life of embedded components. For e.g. In micro-SD card, the life of device is defined in terms of erase cycles. The read, write and erase operations on NAND memory, in solid drives, are performed with application of different voltage levels on memory cell array. This application of voltage causes stress on the cells under operation and the adjacent cells. These stresses cause a permanent and irrecoverable yields in the device and thus limits life time.

[6] Availability is the %age of time over a well-defined period that a system service is available for users.

Availability = MTBF/(MTBF + MTTR)

MTTR = Mean time to recovery.

[3]Instantaneous availability of a component is defined as the probability that the component is functioning properly at time t.

There can be two cases, in case of instantaneous availability:

1. The system/component has not broken till time t.
2. The system was repaired at time x;

$0 < x < t$.

In second case, instantaneous availability, can be computed by summing over the product of reliability for the difference of time between renewal and availability assessment time with the repair rate function.

[8] Scalability is the extent to which the system is capable of growing after its initial deployment. There are system and software factors to consider in scalability. From a system perspective, there are issues of adding further hardware to increase the capacity of the system. In a centralized system, the scope for scalability is limited, such as adding more memory, more disk capacity, or an additional CPU. A distributed system offers much more scope for scalability by adding more nodes to the configuration.

[3]In case of real time embedded system reliability estimation is done for both hardware and firmware.

Because of the complexity involved in such system traditional ways of computing reliability and availability never gives a clean result.

[2]The trend in leveling up the RAS score of the product, measured in terms of CGF grade, provides the reflection of emphasis on RAS laid over by the product company. If we have the right methodologies to decompose and determine the maturity level of the underpinning factors of RAS, from firmware origination point, then the RAS growth can be tracked and visualized easily. Knowledge of the firmware enablers and underpinnings will help in identifying the gaps or holes which is contributing to RAS dip. Once the factor is identified then strategic action can be taken to level-up that factor contributing to RAS negative spike.

[3]The next set of challenge lies in maintaining the highest RAS maturity level/score, once it is achieved.Lack of non-recurring RAS profiling methods may not guarantee firmware's dwell time on an achieved maturity level. Thus, Carrier grading and RAS evaluation shall be a continuous process repeated in conjunction with firmware deliveries.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

The textbook definitions of RAS are intricate and takes an involved approach to evaluate it.

The paper introduces an empirical approach to determine carrier grade category of firmware and thus helps evaluate RAS and classify the carrier grade firmware.Carrier grade maturity level/grade would also help in making choice between two firmware solution. Consider a situation where in we are interested in evaluating two RTOS component for a new embedded platform. Carrier grade maturity level/grade can be added as a parameter to the typical RTOS comparison matrix.

Carrier grade firmware maturity level belongs to below set:

{
NO_CARRIER_GRADE (Grade0/G0),
EMERGE_CARRIER_GRADE (Grade1/G1),
INTERIM_CARRIER_GRADE(Grade2/G2)
MATURE_CARIER_GRADE(Grade3/G3)
}

In short, it can be written as:

$CGF \in \{G0, G1, G2, G3\}$. Below section discusses the details of each of above-mentioned gradelevels.

II. EVALUATING CARRIER GRADE FIRMWARE

This paper discusses a different approachfor evaluating RAS and grade associated with a firmware under evaluation. This approach is based on a set of area of assessment (AoA):

AoA = {
Language,
Object Management,
Task Decomposition,
Critical Sections,
Scheduling,
Communication Protocols,
Fault Safety,
Fault Recovery,
Overload,
Processor failures,
Operability,
Interoperability,
Bootup and upgrade(B&U),
Debugging,
HW Interfacing,
Connectivity
}

If CGS denotes carrier grade software then: $|AoA|_{CGF} > |AoA|_{CGS}$ .The longer AoA list for CGF is due to the fact that there are more variables in CGF related products as compared to CGS products.

Every AoA item has a specified poset of level depending on a bounded criterion.

$$\forall(i \in AoA \ \wedge \ t \in time)\exists Levelj; \quad j \in \mathbb{N}.$$

The CGF is evaluated based on cartesian product of Level(AoA_i) at any given point of time. The CGF qualitatively appears as a total order relation set of levels(denoted by $L_{i;}\, j \in \mathbb{N}$) of AoA_isuch that:

(Let L( ) be the operator which generates level for an AoA)

L(Language)$\in \{L0,\ L1,\ L2\}$ set.
L(Object Management) $\in \{L0,\ L1,\ L2,\ L3\}$set.
L(Task Decomposition) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Critical Sections) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Scheduling) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Comm Protocols) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Fault Safety) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Fault Recovery) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Overload) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Processor failures) $\in \{L0,\ L1,\ L2,\}$ set.
L(Operability) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Interoperability) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(B&U) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Debugging) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(HW interfacing) $\in \{L0,\ L1,\ L2,\ L3\}$ set.
L(Connectivity) $\in \{L0,\ L1,\ L2,\ L3\}$ set.

Language (Lang):

The language of firmware development shall support OOAD ideas and concepts.

$\nexists(OOAD) \rightarrow (L(Language)) == L0$
$\exists Partial\ (OOAD) \rightarrow (L(Language)) == L1$
$\exists N\&S\ (OOAD) \rightarrow (L(Language)) == L2$

N&S in above logical statement means necessary and sufficient. C and assembly language are the major choice of development for firmware, for small to mid-scale embedded systems. And Embedded C++ is choice for mid-scale to large-scale embedded systems. C supports some of the essential OOAD concepts and much can be achieved. However, it is hard to say that a complete OOAD based language like C++ would replace C and assembly based procedural implementation, from embedded systems-based products.

C++ has inherent capability to support scalability without code duplication and obfuscation. The exception handling mechanism supported by this language helps in achieving reliability and serves as a start point for recovery and thus exception handling support becomes a multiplier of availability attribute of the system too.

Object Management (Obj Mgmt):

This AoA is associated with heap and memory block management in firmware.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

$$\exists(MemoryLeak \wedge HeapExhaust \wedge HeapTramplers \wedge Fragmentation) \rightarrow (L(Obj\ Mgmt)) == L0$$

$$\exists(HeapExhaust \wedge HeapTramplers \wedge Fragmentation) \rightarrow ((L(Obj\ Mgmt)) == L1)$$

$$\exists(HeapTramplers \wedge Fragmentation) \rightarrow (L(Obj\ Mgmt)) == L2$$

$$\exists(\downarrow Fragmentation) \rightarrow (L(Obj\ Mgmt)) == L3$$

Generally, if the embedded systems are not too small-scale then there exists a high chance of dynamic memory management and so does the chance of having issues with that. Above logical statements show the conditions to decide the levels with respect to object management.
An inefficient object management will impede high CGF grade achievement and thus it will stand as an impediment for a better RAS score too.

Task Decomposition(Task Decomp):

$$\exists(No\ task \wedge Superloop \wedge Bidirectional\ coupling) \rightarrow (L(Task\ Decomp)) == L0)$$

$$\exists((Coroutines\ V\ ProtoThreads) \wedge \downarrow Bidirectional\ Coupling) \rightarrow (L(Task\ Decomp)) == L1)$$

$$\exists(Tasking \wedge \downarrow Stack\ ripping \wedge \downarrow Bidirectional\ Coupling) \rightarrow (L(Task\ Decomp)) == L2)$$

$$\exists(Tasking) \wedge \nexists(Stack\ ripping) \wedge \nexists(Bidirectional\ Coupling) \rightarrow (L(Task\ Decomp)) == L3)$$

[8] There exists a number of criteria to decompose and structure task groups:
-I/O task structuring criteria
-Internal task structuring criteria
-Task priority criteria
-Task clustering criteria

For achieving CGF highest grade, these criteria shall be met.Task based structural definition of a system is the best way to implement single responsibility design principle. Though tasking helps achieve better modularity and maintainability, if the task decomposition is not properly done then it could be a deterrent factor in achieving CGF. Proper task structuring is a key factor to achieve better scalability.

Critical Sections (CS):

Critical section handling and management is extremely important aspect, in concurrent systems. Critical section is a block of code which needs an exclusive access to the resources that it would access, in the block of code, marked as critical. Otherwise, the state of the data being operated cannot be guaranteed due to race of access over it.

$$\nexists(CS \wedge (system == Asynchronous)) \rightarrow \big(L(CS)\big) == L0$$

$$\exists(CS \wedge Span(CS) > 100\ LOC) \rightarrow \big(L(CS)\big) == L1$$

$$\exists(N\&S\ (CS) \wedge Span(CS) < 100\ LOC) \rightarrow (L(CS)) == L2$$

Critical sections are implemented by:

a. Disabling interrupts before entering CS section and restoring interrupts post CS execution.
b. Locking task pre-emption
c. By introducing resource monitors which in turn uses semaphores with a wait queue.
d. In multicore systems, spinlocks are also usedto gain mutual exclusive access of resources.

This is an extremely important AoA that would impact CGF and RAS score.

Scheduling:

Generally, real time systems perform multiple jobs. Each job needs different handling based on its timing constraints and timeliness requirements. There can be different ways to implement the jobs in order to respect their timeliness requirements. For e.g.

A. Foreground-Background model
B. Scheduler based

$$\exists(Pre-emptive\ scheduling \wedge chance\ of\ starvation) \rightarrow \big(L(Scheduling)\big) == L0$$

$$\exists\begin{pmatrix} Pre-emptive\ scheduling \wedge \\ Round-robinscheduling \wedge \downarrow \\ chance\ of\ starvation \end{pmatrix} \rightarrow \big(L(Scheduling)\big) == L1$$

$$\exists(L1 \wedge Fair-scheduling) \rightarrow \big(L(Scheduling)\big) == L2$$

$$\exists(L2 \wedge Protected\ task\ memory\ spcae) \rightarrow \big(L(Scheduling)\big) == L3$$

The scheduler's key performance indices are the context switching overheads, memory foot print and processor utilization. MMU-aware scheduler which protects the out of schedule task from possible corruptions; will level-up grade of firmware. Systems that have multiple schedulers, and selects scheduler dynamically, can have better RAS score.

Communication Protocols(CP):

Communication protocol plays pivotal role in connectivity of the system. It is important for OAM&P procedures too. Staying connected always reliably, is a much-sought requirement therefore.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

$$\nexists(Asynchronous\ I/O\ \wedge\ prioritization\ ) \rightarrow \big(L(CP)\big) == L0$$
$$\exists(Asynchronous\ Input-Output \wedge State\ machine \wedge Structure\ messaging) \rightarrow \big(L(CP)\big) == L1$$
$$\exists(L1\ \wedge Relaible\ Delivery) \rightarrow \big(L(CP)\big) == L2$$
$$\exists(L2\ \wedge Message\ Bundling) \rightarrow \big(L(CP)\big) == L3$$

In order to achieve better CGF grade and RAS score, CP shall be reliable. It shall incur less overhead on communication bandwidth utilization for user datagram transport. Stateless communication is desired but if reliability is not guaranteed then CP should be either custom developed or shall be picked off-the-shelf with due consideration on session initialization and configuration latencies. The asynchronous events in CP shall be limited as it causes stack ripping and increases complexity of the system design.

Fault Safety (FS):

High graded CFG firmware shall have detailed exception handling for faults that can occur within the firmware or could be caused due to undetermined sequence of external events that impacts the system.

$$\nexists(Fault\ containment\ techniques) \rightarrow \big(L(FS)\big) == L0$$
$$\exists(Defensive\ Coding \wedge Low\ order\ page\ protection) \rightarrow \big(L(FS)\big) == L1$$
$$\exists(L1 \wedge Stack\ overflow\ protection) \rightarrow (L(FS)) == L2$$
$$\exists(L2 \wedge low\ order\ page\ protection) \rightarrow (L(FS)) == L3$$

In order to improve performance, now-a-days, the RAM sizes are increased multi fold and the data accessed during execution is also contained in RAM. Since the RAM is accessible by any entity until MMU is properly configured and handles read and write access properly. Tramplers in the system can cause random corruption by accessing their arrays out of bound or by using a pointer that is obtained with incorrect arithmetic on addresses.

Thus, a safety net shall be provisioned to catch and handle exceptions and restore normal firmware flow.

Fault Recovery (FR):

The firmware fault recovery system shall comprise of an early fault detection sub-system and then an escalated fault recovery mechanism. An unrecovered firmware shall try recovery from light-weighted possible recovery procedure to heavy weight recovery including hardware resets. Heavy and light weight in this case implies the execution time and dedicated resources needed to recover.

$$\exists(Reboot \wedge Reinit) \rightarrow \big(L(FR)\big) == L0$$
$$\exists(Init\ Framework \wedge Watchdog) \rightarrow \big(L(FR)\big) == L1$$
$$\exists(L1 \wedge HealthMonitor) \rightarrow \big(L(FR)\big) == L2$$
$$\exists(L2 \wedge Escalating\ Restart \wedge binary\ version\ rollback) \rightarrow \big(L(FR)\big) == L3$$

Thermal management Units (TMUs), Brown out detector(BOD), Watchdog(WDG) and a monitoring processor core can help detect faults very early and trigger recovery. Fault recovery shall take least possible time as it directly impacts availability.

In another approach, with multicore capability, system can enter a reduced functioning mode and continue recovery while limited availability can still be guaranteed.

Shadow drivers, system health monitors and anomaly detectors are few components which fills the void in fault recovery aspect of the system.

Overload:

There shall be a well specified idle time such that a system should mostly enter idle with in that specified time. Any miss in this respect, would mean that the system is overloaded with lot of task. In this situation, some of the high priority tasks hogs the system.

$$\exists Thrashing \rightarrow \big(L(Overload)\big) == L0$$
$$\exists(Deterministic\ job\ completion \wedge discard\ new\ work) \rightarrow \big(L(Overload)\big) == L1$$
$$\exists(L1 \wedge babbling\ component\ handling) \rightarrow \big(L(Overload)\big) == L2$$
$$\exists(L2 \wedge Fine\ throttling\ mechanism) \rightarrow \big(L(Overload)\big) == L3$$

This is an important aspect for ensuring high RAS score. Any easily and frequently overloaded system cannot help achieve carrier grade and thus would demand refactoring.

Processor Failures (PF):

Processors these days supports multiple operating performance points(OPP) and so is their variation on loads. Processors within the system shall be regressed under different scenarios and for all supported and used OPPs. Any hardware

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

assumption or miss, may cause processor to fail during execution.

$$\nexists(Backup\ processing\ element) \rightarrow \big(L(PF)\big) = = L0$$

$$\exists(Load\ sharing\ \wedge cold\ standby\ processors) \rightarrow \big(L(PF)\big) == L1$$

$$\exists(L1 \wedge hot\ standby\ processors\ ) \rightarrow (L(PF)) = = L2$$

During sleep and wakeup cycles, for power saving, a system with multiple hot-pluggable processors can cause SCR latch-up in case the dynamic loading and all component connection to voltage and ground rails are not designed to cope and counter against such phenomenon. The processor sub-system would not resume normal operation, in this case. Power consumption is now a much-desired feature and thus the RAS score and CGF matters a lot in this regard.

Operability:

This AoAis related to administration services in the system. For e.g., configuration, logging, alarms, profiling and maintenance.

$$\exists(Basic\ administrative\ services) \rightarrow (L(Operability) == L0)$$

$$\exists(Configuration\ \wedge Logging\ \wedge Alarms \wedge Profiling\ \wedge Maintenance) \rightarrow \big(L(Operability)\big) == L1$$

$$\exists\ L1 \wedge Out\ of\ service\ diagnosis \rightarrow (L(Operability)) == L2$$

$$\exists\ L2 \wedge in\ service\ diagnosis \rightarrow (L(Operability)) == L3$$

As this AoA deals with configuration thus it will greatly impact scalability QoS of the system besides factoring availability.

Inter-operability:

When a system becomes a communicating party of another system and together with other connected systems, serves to the user, then in that case interoperability plays a crucial role for RAS.

$$\nexists(Connectivity\ to\ any\ desired\ devices) \rightarrow \big(L(IOP)\big) == L0$$

$$\exists(Connectivity\ to\ a\ set\ of\ desired\ devices) \rightarrow \big(L(IOP)\big) == L1$$

$$\exists(Connectivity\ to\ all\ possible\ desired\ devices) \wedge\ connection\ need\ high\ user\ intervention) \rightarrow \big(L(IOP)\big) == L2$$

$$\exists(Connectivity\ to\ all\ possible\ desired\ devices) \wedge\ no\ user\ intervention) \rightarrow \big(L(IOP)\big) == L3$$

Bluetooth, Wifi, NFC, SIM technology and cellular technologies are some of the areas where interoperability is a major concern and many forums provide platform to test and determine a device's interoperability capabilities.

Boot up And Upgrade(B&U):

A fast, secure and reliable boot-up of firmware and underlying hardware is always desired.

Bootup sequences are typical initialization and configuration set of procedures. Bootup behaviour involves all of the component of the system and shows dynamic behaviour due to device aging factors and environmental conditions.

$$\exists(Boot\ banner\ \wedge \sim failure\ message \wedge Chance\ of\ irrecoverable\ device) \rightarrow \big(L(B\&U)\big) == L0$$

$$\exists(L0\ \wedge Debuggable\ bootloader\ ) \rightarrow \big(L(B\&U)\big) == L1$$

$$\exists(L1\ \wedge Safe \wedge Upgrade\ procedure) \rightarrow \big(L(B\&U)\big) == L2$$

$$\exists(L1\ \wedge Upgrade\ procedure \wedge Firmware\ rollback) \rightarrow \big(L(B\&U)\big) == L3$$

Many security related parameters are loaded in memory. The faulty device isolation, reduced function mode entry, debuggability and high reliability is much desired.Similarly, firmware upgrade shall be very deterministic and shall be quick and shall support rollback mechanism on failure. The device shall recover quickly in case the upgrade fails and should retain usable and original form. Upgrade failure shall be communicated to OAM&P system so that it can be attempted again.

Debugging:

It is extremely important for firmware to support multiple debugging capability such as debug over JTAG, debug over UART/USB, core dumping and remote debugging.

$$\nexists(Debug\ sub-system) \rightarrow \big(L(Debug)\big) == L0$$

$$\exists(Warning\ logs\ \wedge Function\ tracer \wedge Message\ tracer \wedge Core\ dump) \rightarrow \big(L(Debug)\big) == L1$$

$$\exists(L1\ \wedge\ HW\ flight\ recorder \wedge Multi\ HW\ debug\ channels) \rightarrow \big(L(Debug)\big) == L2$$

$$\exists(L2\ \wedge\ trace\ buffer) \rightarrow \big(L(Debug)\big) == L3$$

If, on product failure, firmware and hardware state could be obtained using supported debug capabilities, such that root cause of the failure could be reasoned out, then the probability of rapid convergence to maximum CGF grade, will increase.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

HW interfacing (HWI):

In modern times, many semiconductor companies have established hardware firmware co-design practices. Engineers from these two different disciplines meet together to understand product and underlying system specification and requirements and co-design to achieve optimum solution.

$$\exists(Hardwired\ device\ configuration \wedge Limted\ on - chip\ device\ set) \rightarrow \big(L(HWI)\big) == L0$$

$$\exists(Firmware\ configurable\ devices \wedge less\ SFR \wedge high\ special\ instructions) \rightarrow \big(L(HWI)\big) == L1$$

$$\exists(Firmware\ configurable\ devices \wedge N\&S(SFR) \wedge N\&S(\ special\ instructions) \wedge HW\ accelerators \wedge HW\ accelerators \wedge Device\ redundancy) \rightarrow \big(L(HWI)\big) == L2$$

$$\exists(L2 \wedge Coprocessor \wedge HW\ monitors \wedge HW\ loggers) \rightarrow \big(L(HWI)\big) = = L3$$

It is desirable to have a smaller number of related registers that need to be programmed to perform a single job. Hardware shall provide as many chicken bits as possible so that hardware features can be enabled disabled orthogonally. Hardware shall also provide interfaces for recovery such as soft reset bit or warm reset bits. At any point of time, it is desirable for firmware to peek into hardware FSM and retrieve the status of ongoing processing in hardware. The hardware shall incur specified latency with known margins for the processing it offers. The register organization shall be grouped under different functional blocks, The documentation and naming conventions shall be adopted with ease of usage in mind. The default values of registers on PoR shall be selected in such a way that for a typical configuration the device shall be ready of operation with very few firmware settings.

III. PROCEDURE TO QUANTIFY AOA LEVELS AND CARRIER GRADE FOR FIRMWARE

In this section, the procedure to evaluate the levels of AoA and thus CGF grading is explained, along with couple of possible outcomes in the form of graph. Consider there is a dedicated development team which is focussed only on CGF improvement. The team members are highly skilled and have spent around 8 to 10 years in embedded development then a year-on-year ramp can be achieved for many of the AoA. In case the team has

Connectivity:

In this era, where massive integration of technologies is in vogue, the systems and products need to have multiple connectivity options to talk to other devices both wired and wirelessly.

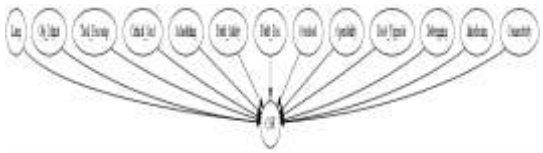
$$\nexists(No\ connectivity) \rightarrow \big(L(Connectivity)\big) = = L0$$

$$\exists(Limited\ single\ channel\ connectivity) \rightarrow \big(L(Connectivity)\big) == L1$$

$$\exists\begin{pmatrix}Limited\ configurable\ multi\\ channel\ connectivity\end{pmatrix} \rightarrow \big(L(Connectivity)\big) == L2$$

$$\exists\begin{pmatrix}Auto\ configurable\ secure\\ multi\ channel\ connectivity\end{pmatrix} \rightarrow \big(\mathrm{L(Connectivity)}\big) == \mathrm{L3}$$

A remarkable RAS score and CGF upgrade can only be achieved if there existalternate channels to connect and communicate without impacting user experience. Another desired attribute in terms of connectivity, is automatic and seamless connection with other device, without much user intervention.



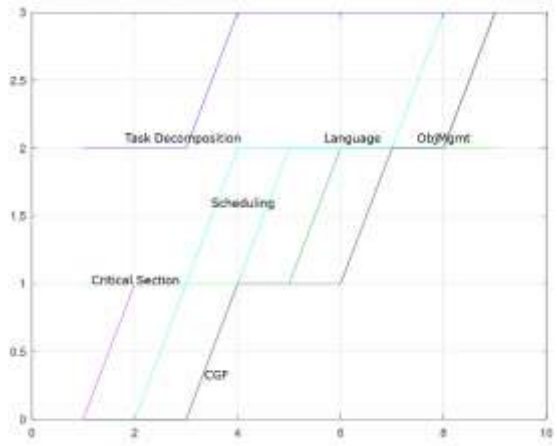
Factors that contribute to CGF

mix experienced members and has shared duties like maintenance and testing, in that case it is likely that the ramp will not be consistent on yearly basis. Some of the AoA will roll down and thus pull down CGF grade too.

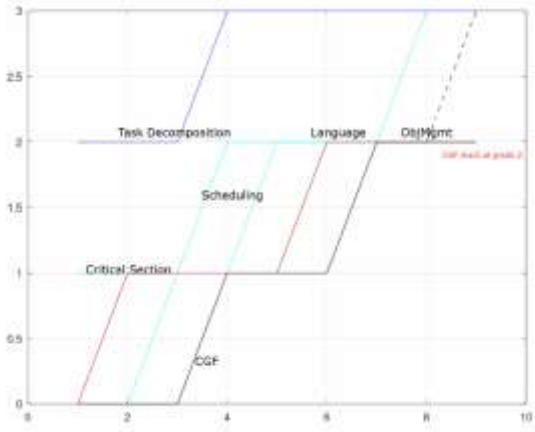
In any case, the AoA shall be evaluated periodically, every year, based on criteria mentioned in above section. At the end of evaluation cycle, the grade of CGF is assigned as the level achieved by all AoA in that evaluation cycle.

The tables shared is based on past experience of working in this area.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252



Consistent CGF growth



Below table summarizes the growth pattern of AoAs:

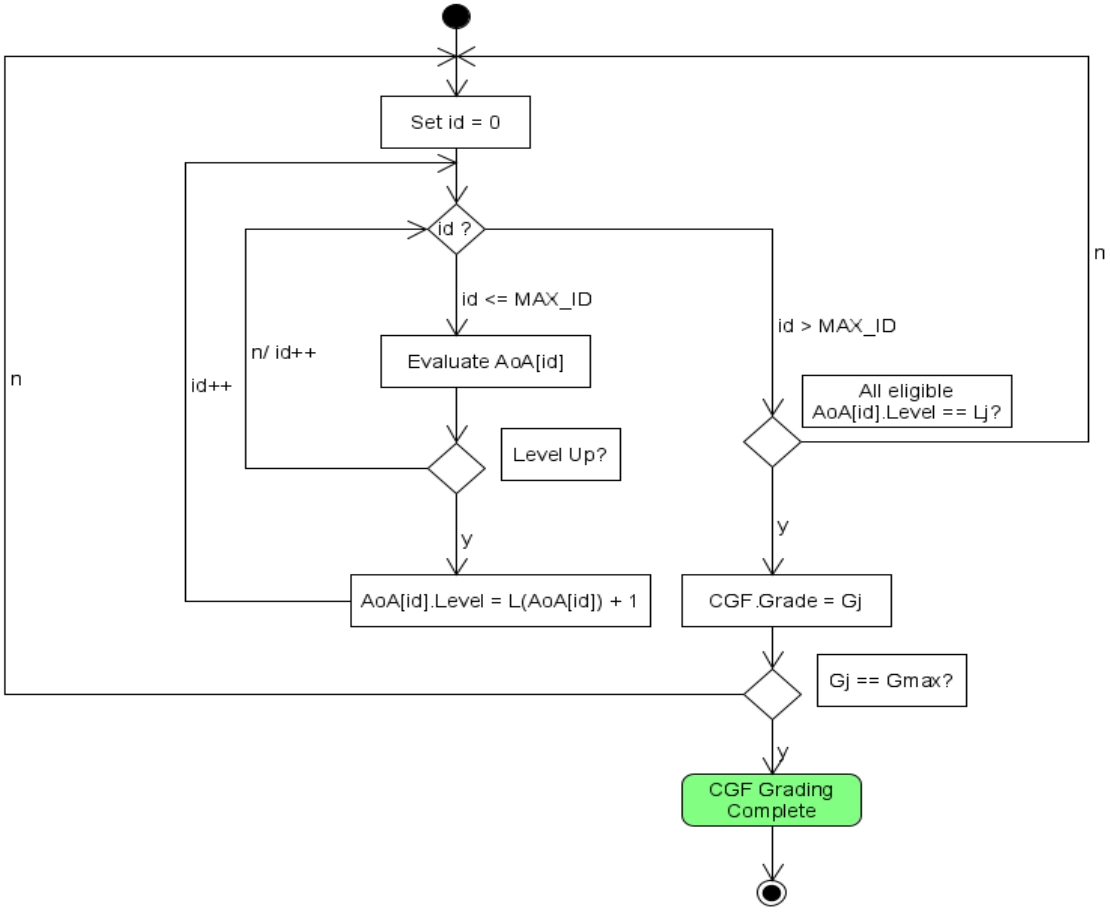
Year	1	2	3	4	5	6	7	8	9
AoA /Levels->									
Language	L0	L0	L1	L1	L2	L2	L2	L3	L3
Object Management	L0	L1	L1	L1	L1	L2	L2	L2	L3
Task Decomposition	L2	L2	L2	L3	L3	L3	L3	L3	L3
Critical Sections	L1	L1	L1	L1	L1	L2	L2	L2	L2
Scheduling	L1	L1	L1	L2	L2	L2	L2	L2	L3
Communication Protocols	L1	L1	L2	L2	L2	L2	L2	L2	L3
Fault Safety	L1	L1	L1	L2	L2	L2	L2	L2	L3
Fault Recovery	L1	L1	L1	L1	L1	L2	L2	L2	L3
Overload	L1	L1	L1	L1	L2	L2	L2	L3	L3
Processor failures	L0	L1	L1	L1	L1	L2	L2	L2	L2
Operability	L1	L1	L1	L1	L2	L2	L3	L3	L3
Interoperability	L0	L0	L0	L1	L1	L1	L2	L2	L3
Boot & Upgrade	L1	L1	L1	L2	L2	L2	L3	L3	L3
Debugging	L1	L2	L2	L2	L3	L3	L3	L3	L3
HW interfacing	L1	L1	L1	L2	L2	L3	L3	L3	L3
Connectivity	L1	L1	L2	L2	L2	L2	L2	L3	L3

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 127-136 www.ijaem.net ISSN: 2395-5252

IV. CGF GRADE TO RAS SCORE CONVERSION

A function is feasible which can derive RAS score based on obtained CGF grade and it can be used to compare against the RAS mentioned in firmware requirement specification. A weighted average could be a good function which can map CGF grade to RAS score.
The details of conversion of CGF grade to RAS score lies beyond the scope of this paper.

CGF stuckat grade 2



Flow chart to determine carrier grade of firmware

V. CONCLUSION

The world of electronic consumer products is going to view the product and its underpinnings differently and special attention will be paid for the product RAS evaluation and specification. The growing awareness among user base and the nature of competitive market for consumer electronics goods (with semiconductor supplies in it) will provide extra push to the production houses and besides the QoS and time-to-market trade-offs, companies will have to go extra mile to ensure high RAS score.

CGF evaluation will prove to be a practical and time saving way to quantify and visualize the trend in growth towards RAS score. It will help in checkpointing on different AoA and would help setup a controlled environment for firmware development whose only goal will be to achieve high RAS score and maintain the score by making sure it does not dwindle.

REFERENCES

[1]. Kim H. Pries, Jon M. Quigley; Project Management Of Complex And Embedded Systems
[2]. Jiacun Wang; Real-Time Embedded System
[3]. Kishore S. Trivedi;Probability & Statistics with reliability, queuing and computer science applications
[4]. Philip A. Laplante; Real-Time Systems Design and Analysis
[5]. C.M. Krishna, Kang G. Shin;Real-Time Systems
[6]. Sam Siewert; Real-Time Embedded Systems And Components
[7]. Greg Utas; Robust Communications Software
[8]. Hassan Gomaa; Real-Time Software Designs for Embedded Systems