Best Practice for Automotive Linux Adoption

HISAO MUNAKATA
RENESAS SOLUTIONS CORP
hisao.munakata.vt(at)renesas.com

Disclaimer

Everything I say here is just my opinion and not the opinion of my employer Renesas. As our experience of Linux and other opensource architecture adoption for the automotive products is quite limited, my understanding here might not be correct or missing something. If you have objection to my opinion, please collect me at any time. I appreciate for your permissiveness.

who am I

- Working for Renesas (semiconductor)
  - Over 15 years “real embedded Linux business field” experience
  - Provide “free Linux starter code (BSP)” for our platform
  - Support Linux newbie's but important customer (who asks everything about Linux to us)
  - Over 5 years experience working with the community
  - Linux Foundation CEWG Architecture groupe co-chair
  - We have Linux core technology team inviting some key community developer to put our code into Linux upstream.

We have learned how to work with Linux community

3 topics for today's talk

- High quality automotive produces + High quality Linux code = perfect match?
- Paradigm shift : "Bug free design" to "Fault tolerant design"
- How can you play with source code ?

1st topic for today’s talk

- High quality automotive produces + High quality Linux code = perfect match?
- Paradigm shift : “Bug free design” to “Fault tolerant design”
- How cab you play with source code ?

There are 2 proven high quality stuff

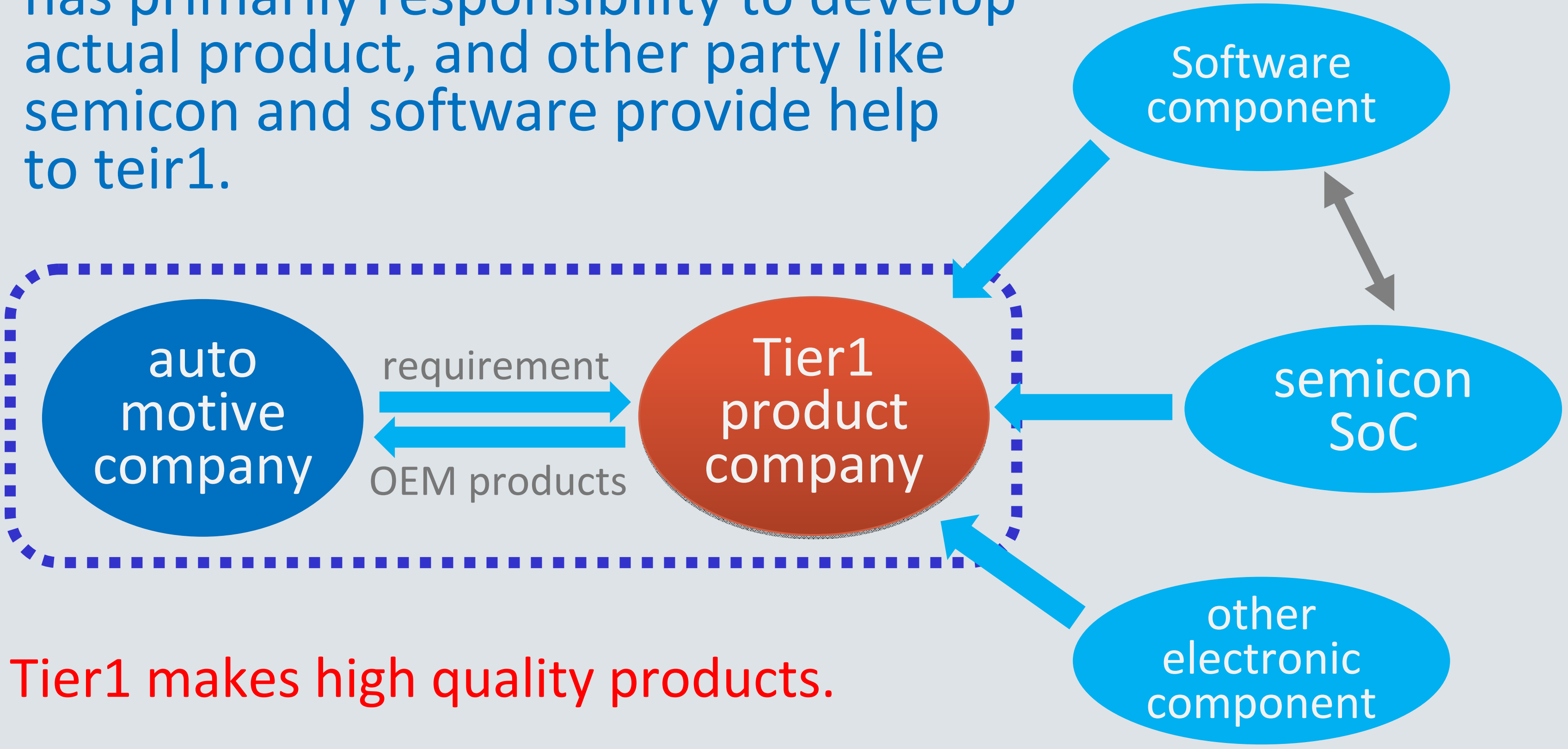
Automotive Electric equipment design

Linux & opensource development

Automotive Electric equipment design

Proven automotive product design scheme

- Established “technology supply chain” that can achieve very high availability system. Tier1 product producer company has primarily responsibility to develop actual product, and other party like semicon and software provide help to teir1.



Tier1 makes high quality products.

Linux & Opensource development

Proven Linux technology and development

- Linux is adopted for various mission-critical infrastructure
  - Securities exchange system
  - Banking system
  - Airline ticket reservation system
  - Super computer
  - Digital TV, HDR (HDD recorder)
  - Smartphone (Android, etc..)

enterprise

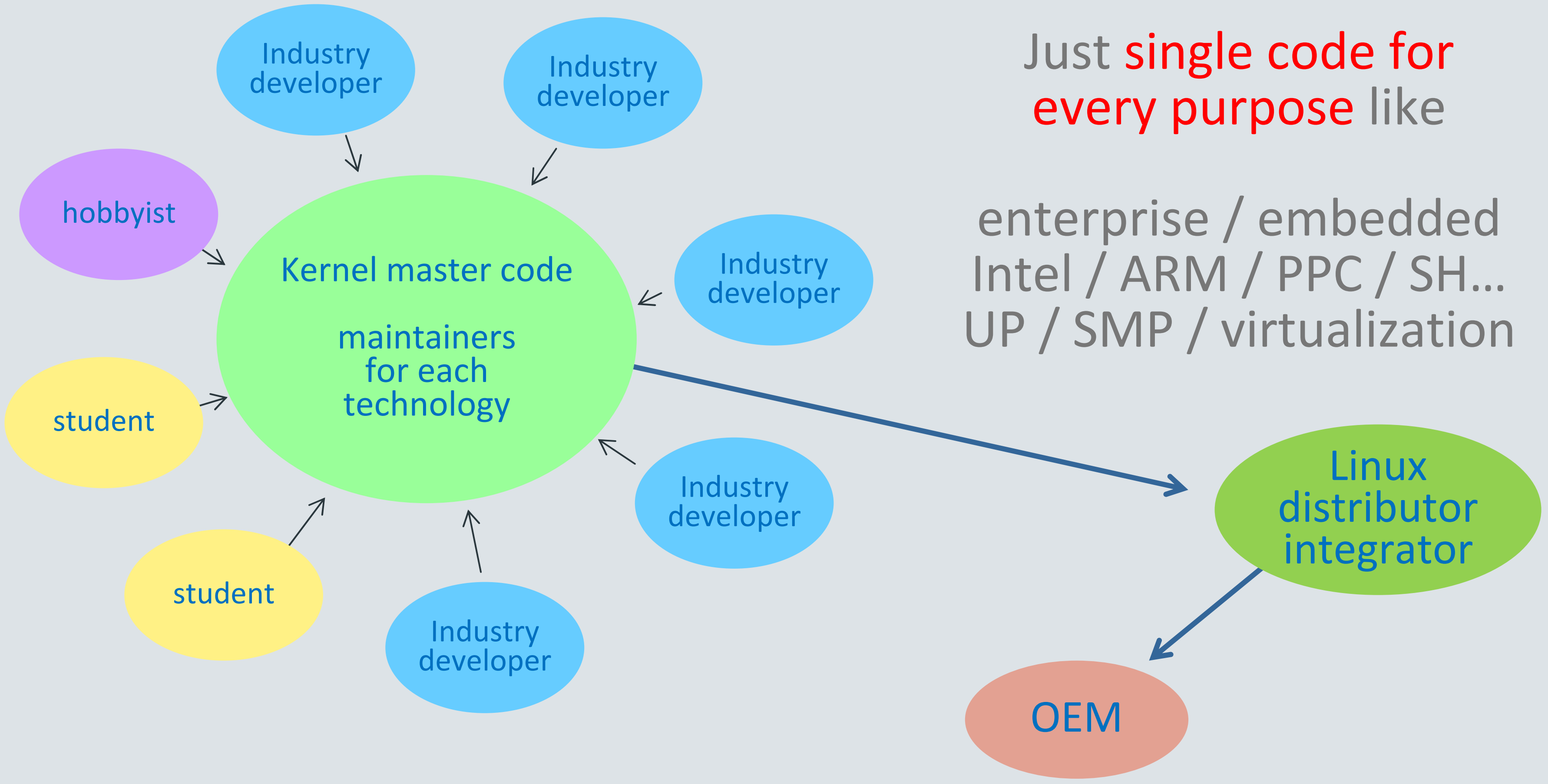
embedded

- Development result of Linux kernel is outstanding
  - Over 15 million lines of code (Linux 3.0)
  - Over 1,300 developers contribute to development
  - New kernel had been released every 3 month.

Kernel development community develops high quality code.

Linux & Opensource development

Linux kernel development & release flow



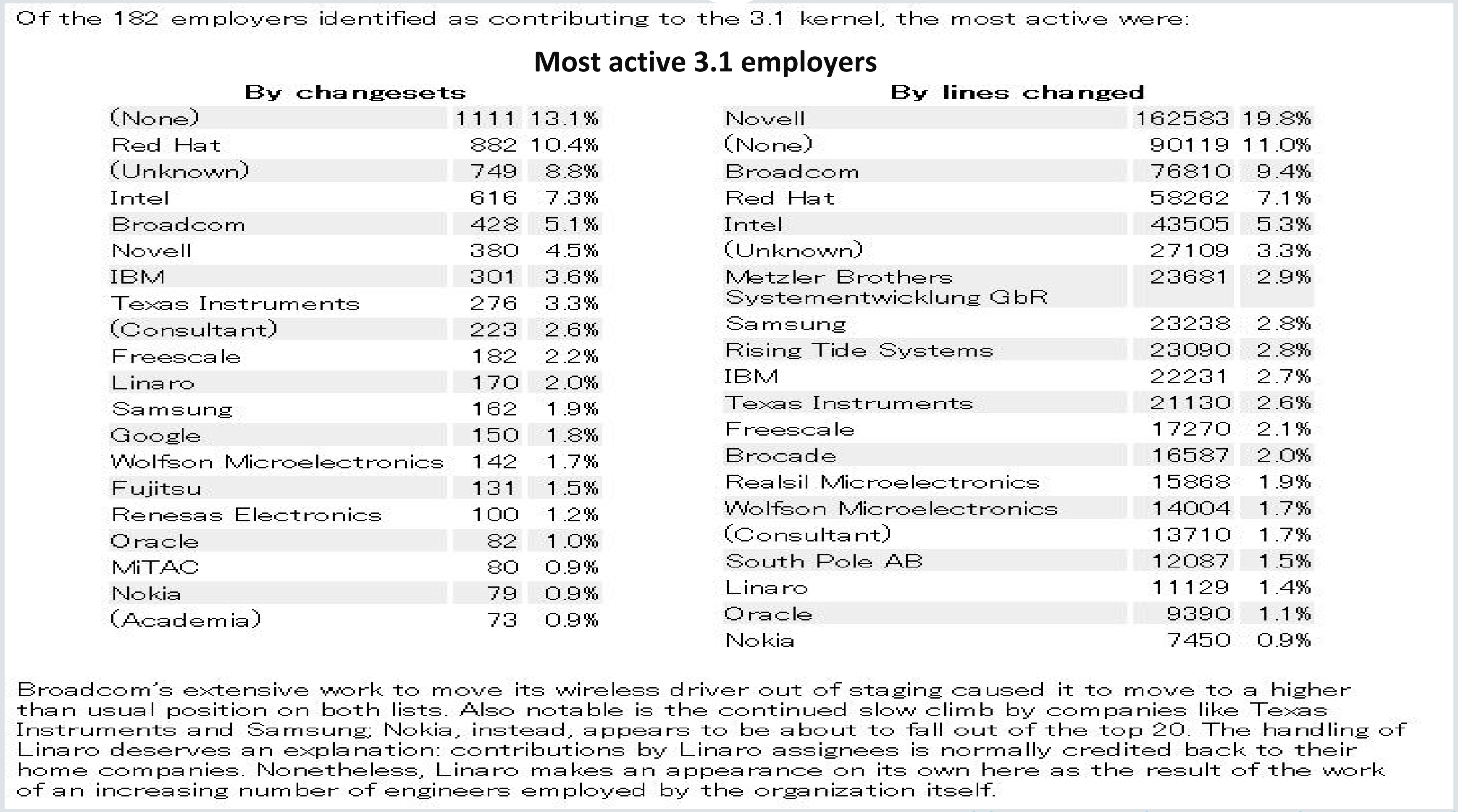
Just single code for every purpose like

enterprise / embedded
Intel / ARM / PPC / SH…
UP / SMP / virtualization

Majority of kernel code are developed by industry professional developer

Linux & Opensource development

Linux kernel 3.1 developer statistics



Of the 182 employers identified as contributing to the 3.1 kernel, the most active were:

Most active 3.1 employers

By changesets

(None)	1111	13.1%
Red Hat	882	10.4%
(Unknown)	749	8.8%
Intel	616	7.3%
Broadcom	428	5.1%
Novell	380	4.5%
IBM	301	3.6%
Texas Instruments	276	3.3%
(Consultant)	223	2.6%
Freescale	182	2.2%
Linaro	170	2.0%
Samsung	162	1.9%
Google	150	1.8%
Wolfson Microelectronics	142	1.7%
Fujitsu	131	1.5%
Renesas Electronics	100	1.2%
Oracle	82	1.0%
MiTAC	80	0.9%
Nokia	79	0.9%
(Academia)	73	0.9%

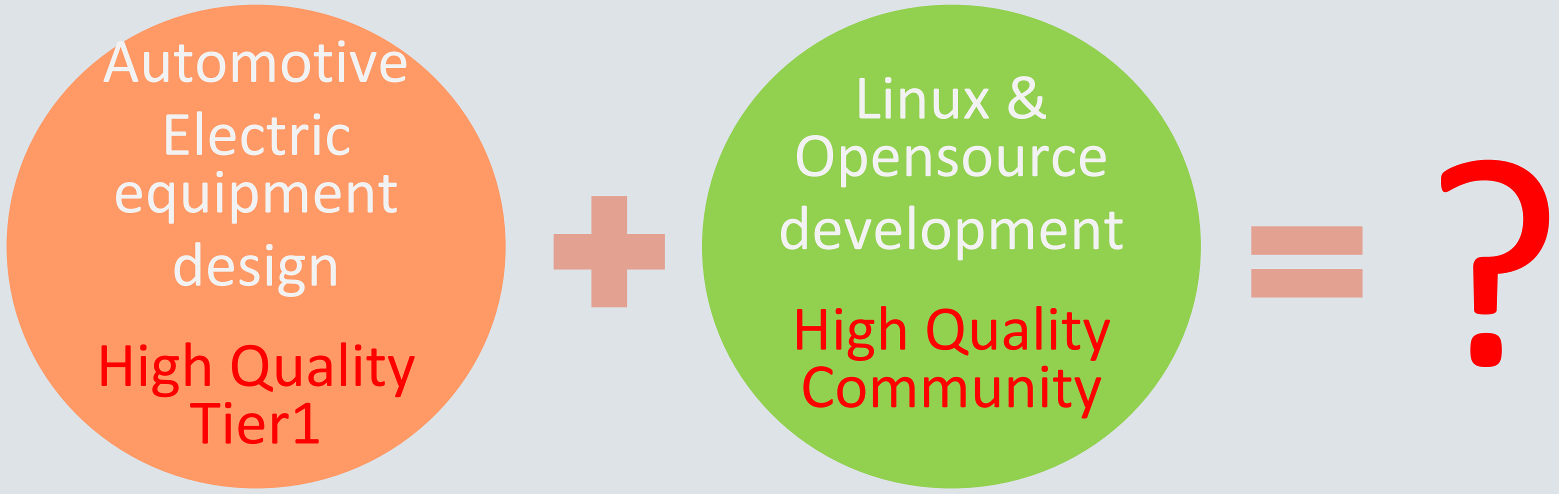
By lines changed

Novell	162583	19.8%
(None)	90119	11.0%
Broadcom	76810	9.4%
Red Hat	58262	7.1%
Intel	43505	5.3%
(Unknown)	27109	3.3%
Metzler Brothers Systementwicklung GbR	23681	2.9%
Samsung	23238	2.8%
Rising Tide Systems	23090	2.8%
IBM	22231	2.7%
Texas Instruments	21130	2.6%
Freescale	17270	2.1%
Brocade	16587	2.0%
Realsil Microelectronics	15868	1.9%
Wolfson Microelectronics	14004	1.7%
(Consultant)	13710	1.7%
South Pole AB	12087	1.5%
Linaro	11129	1.4%
Oracle	9390	1.1%
Nokia	7450	0.9%

Broadcom's extensive work to move its wireless driver out of staging caused it to move to a higher than usual position on both lists. Also notable is the continued slow climb by companies like Texas Instruments and Samsung; Nokia, instead, appears to be about to fall out of the top 20. The handling of Linaro deserves an explanation: contributions by Linaro assignees is normally credited back to their home companies. Nonetheless, Linaro makes an appearance on its own here as the result of the work of an increasing number of engineers employed by the organization itself.

http://lwn.net/Articles/460597/

Question ?


Automotive Electric equipment design
High Quality Tier1
Linux & Opensource development
High Quality Community

Can Tier1 utilize current Linux asset by themselves ?

Automotive Electric equipment design

Automotive system design assumption

Software component

Code is fixed and enough solid

↓

Semiconductor / Software provider

Can provide fully validated bug-free solution

↓

Tier1 OEM company

Fixed use case defined by auto company

Linux upstream development philosophy

Always chasing moving target

- huge code size [over 15 million lines]
- very high complexity
- big [around 125k line] code change at every 3 month new kernel release
- adopt new technology, new device support
- not enough system-wide validation

Speed is everything for keeping evolution

Automotive Electric equipment design

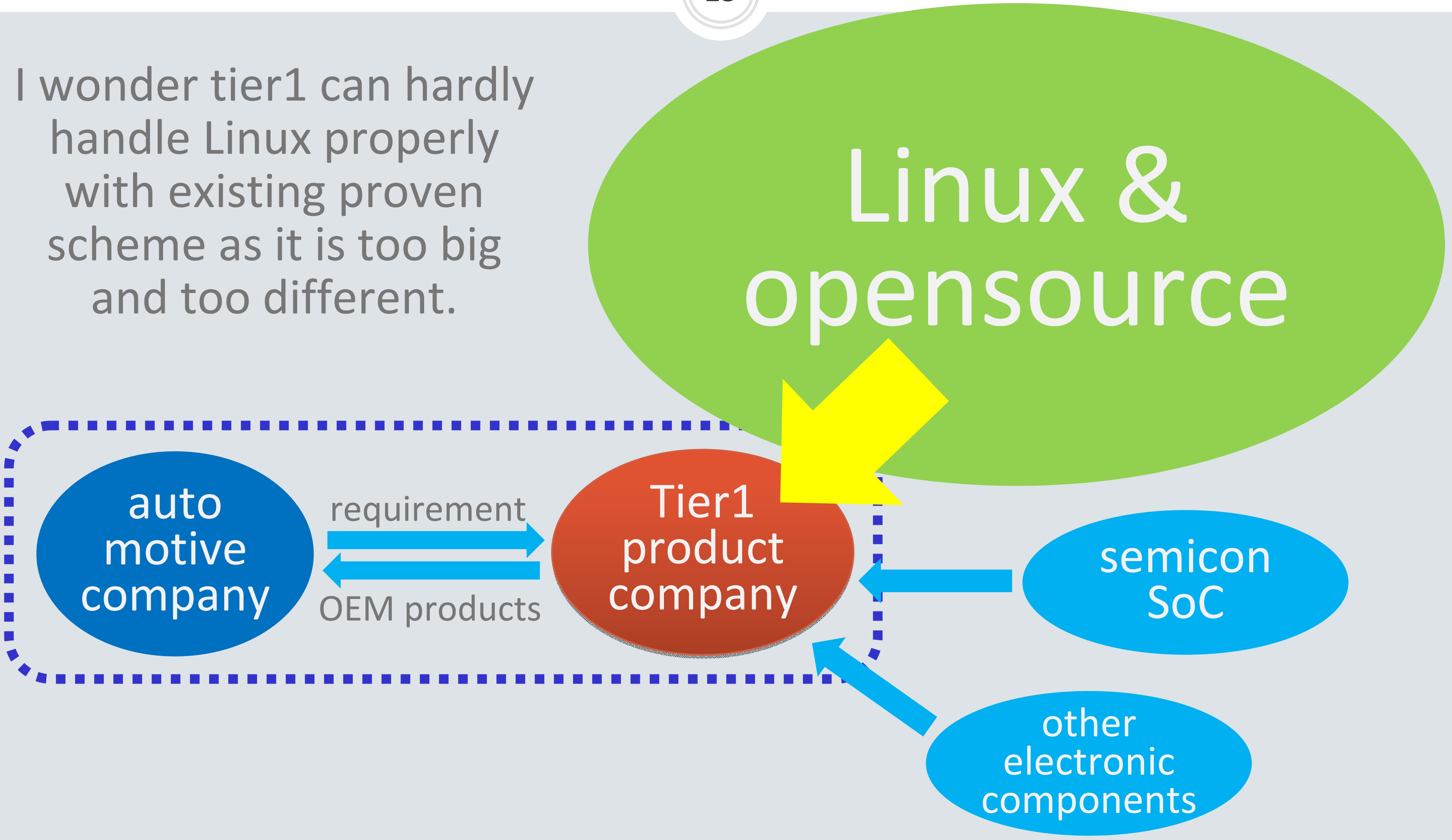
Technology contrast

Linux & Opensource development

RTOS	Linux
deterministic	heuristics
static	dynamic
fixed resource allocation	flexible resource allocation
predictable behavior	runtime coordination
write real address	abstraction
company product	community work

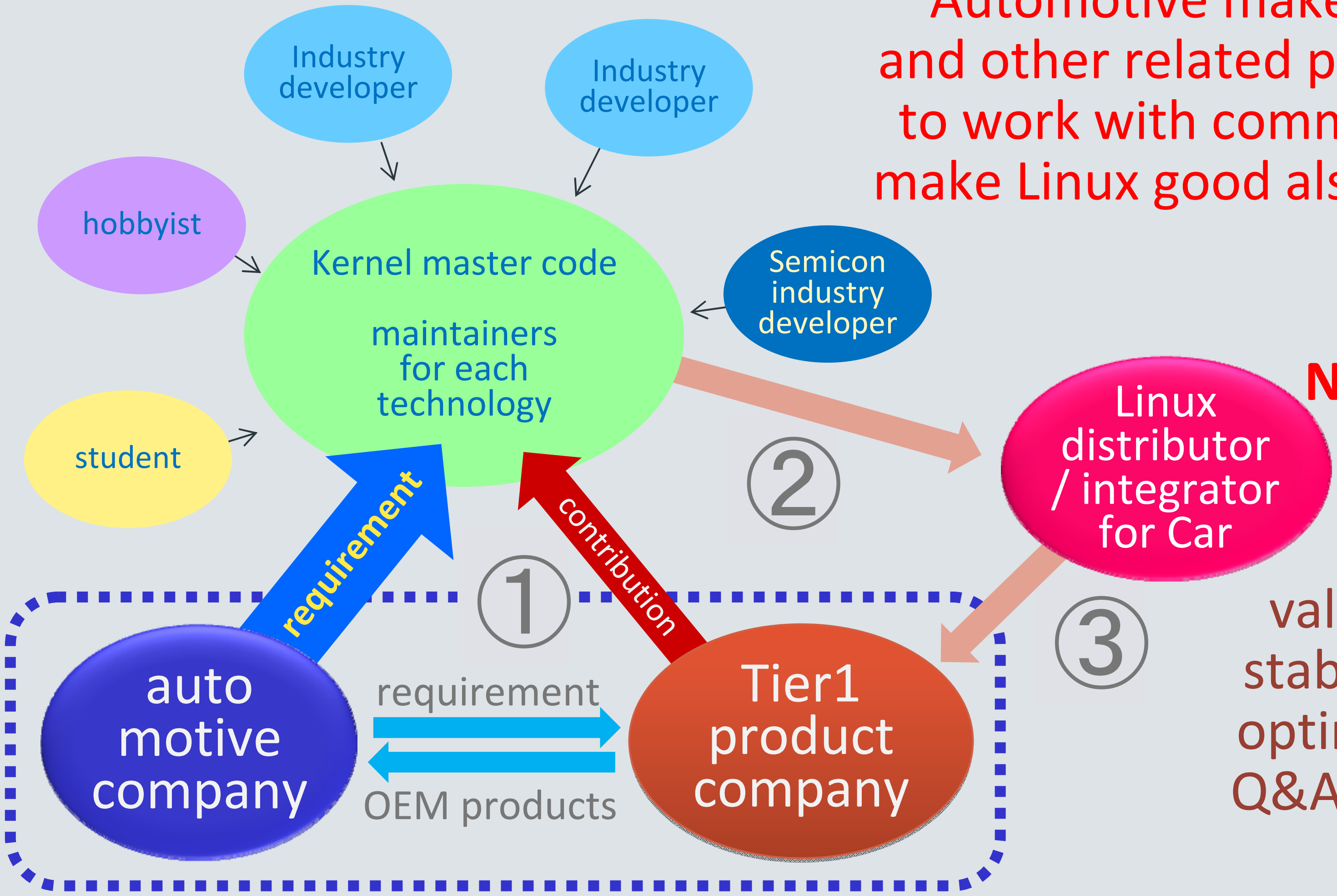
Will existing Tier1 model work?

I wonder tier1 can hardly handle Linux properly with existing proven scheme as it is too big and too different.



New eco-system proposal to utilize Linux on car

Automotive maker, tier1 and other related party need to work with community to make Linux good also for Car.



validation
stabilization
optimization
Q&A support

2nd topic for today's talk

- High quality automotive produces + High quality Linux code = perfect match?
- Paradigm shift : "Bug free design" to "Fault tolerant design"
- How can you play with source code ?

Definition of “Quality” (by ISO)

The quality of something can be determined by comparing a set of inherent characteristics with a set of requirements. If those inherent characteristics meet all requirements, high or excellent quality is achieved. If those characteristics do not meet all requirements, a low or poor level of quality is achieved.

Quality is, therefore, a question of degree. As a result, the central quality question is: How well does this set of inherent characteristics comply with this set of requirements? In short, the quality of something depends on a set of inherent characteristics and a set of requirements and how well the former complies with the latter.

According to this definition, quality is a relative concept. By linking quality to requirements, ISO 9000 argues that the quality of something cannot be established in a vacuum. ***Quality is always relative to a set of requirements.***

$$\text{Quality} = \frac{\text{Validation}}{\text{Requirements}}$$

http://www.praxiom.com/iso-definition.htm#Quality

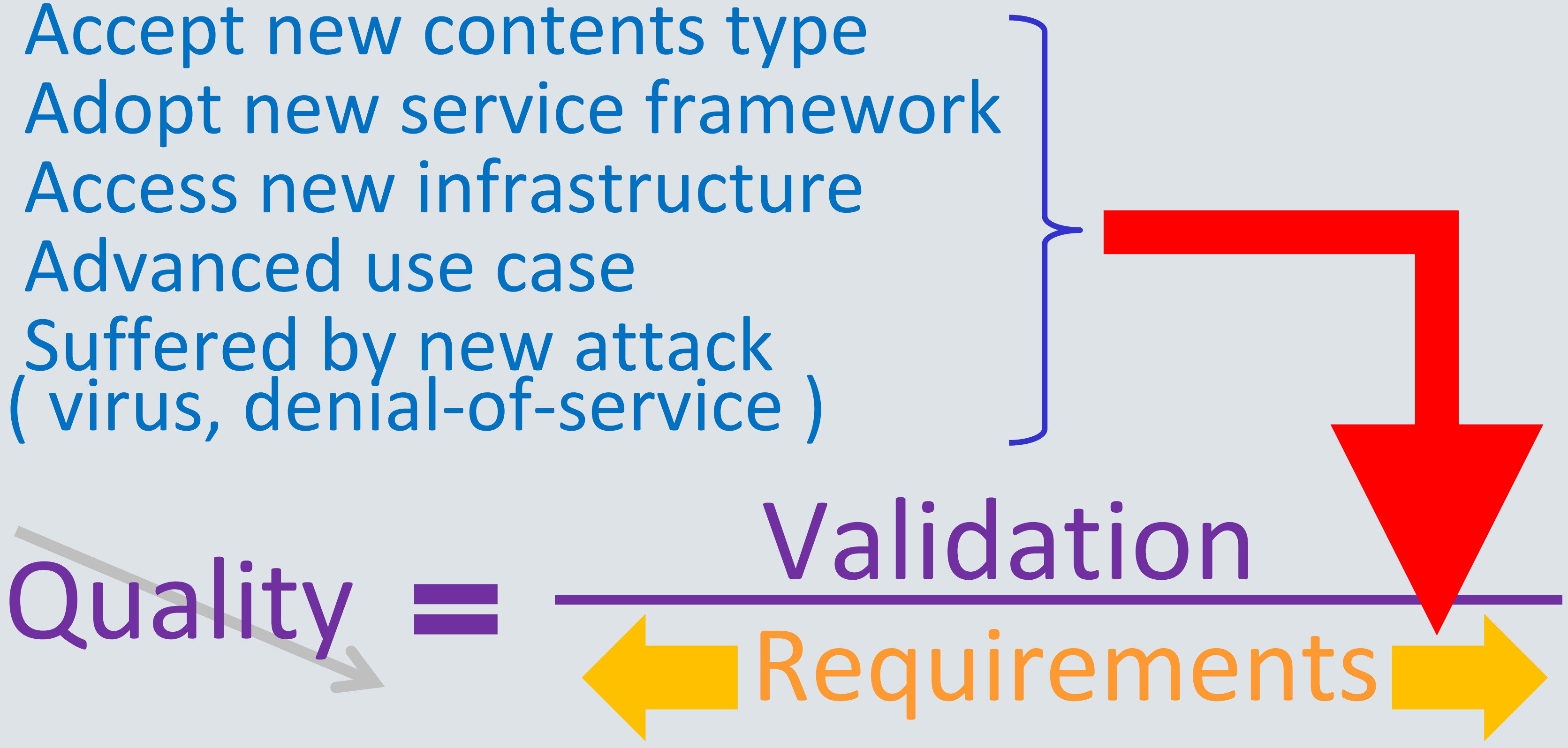
BTW, why Linux for automotive and who ?

- **Linux = Open solution platform for car**
  - global asset
    - can utilize w/w programmer resources
    - public source code (easy to share)
  - rich connectivity
    - Network protocol
    - Support various storage device
    - Cloud service connection
  - good security fix management
  - social service platform

Car producer's new demands

“Openness” of modern service platform may cause bloat of new requirements

- Accept new contents type
- Adopt new service framework
- Access new infrastructure
- Advanced use case
- Suffered by new attack (virus, denial-of-service)



Validation might not catch up with ***bloated requirements***

We have experienced serious system failure

Cell-phone network down

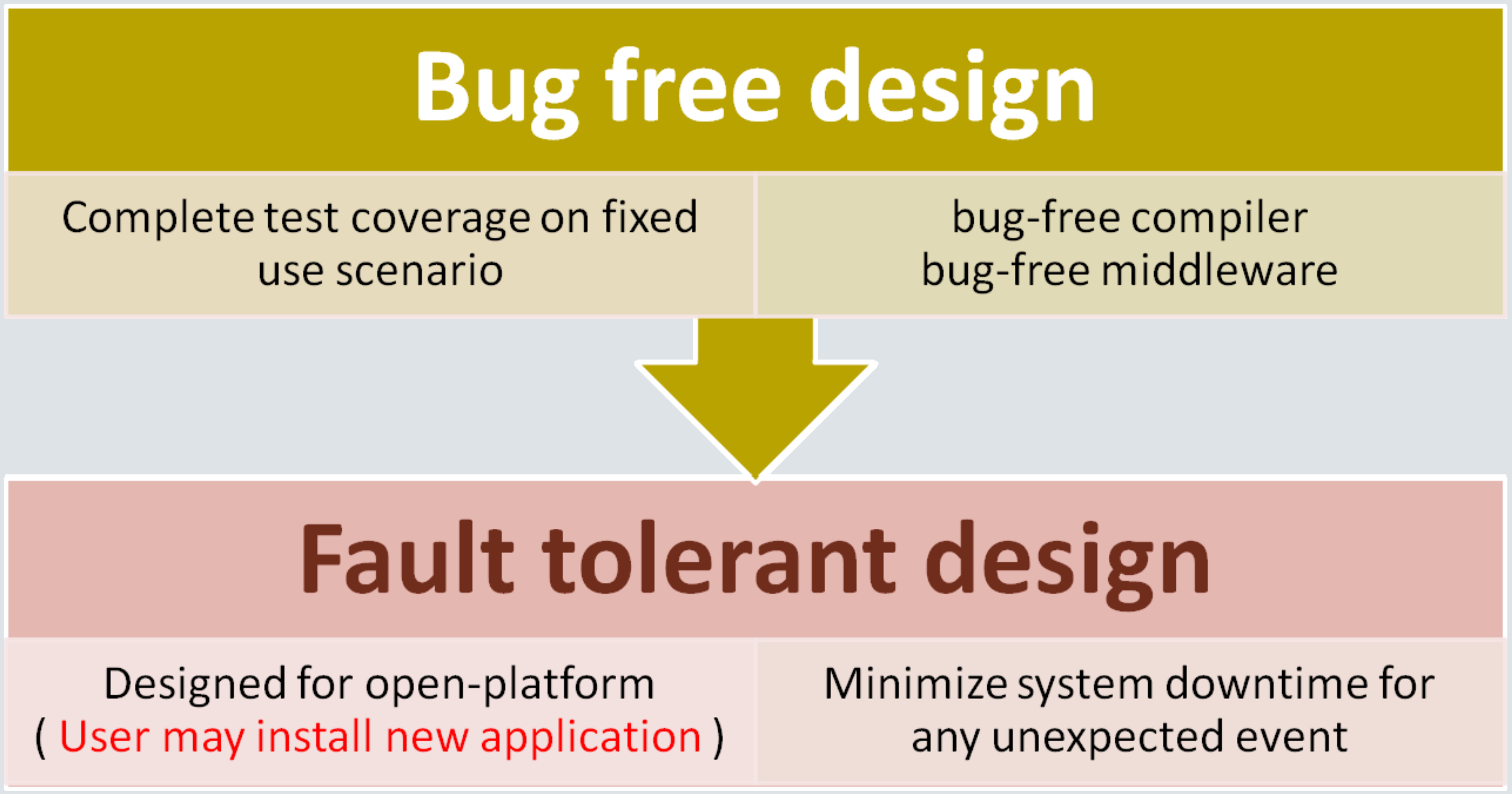
Banking ATM system down

Airline ticketing system down

Train traffic control system malfunction

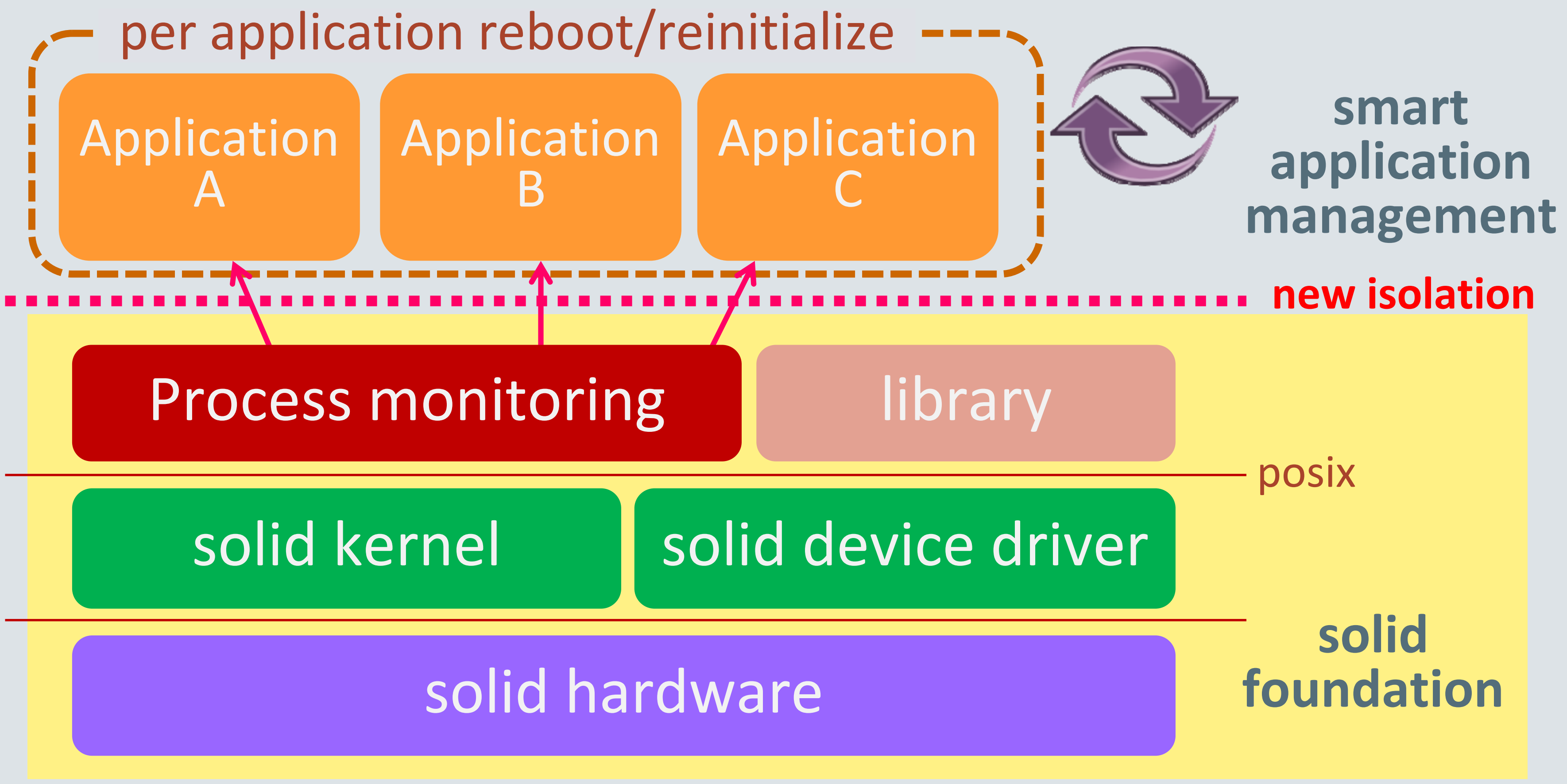
How can we minimize whole system down-time

Paradigm shift for open-platform redundancy


Bug free design
Complete test coverage on fixed use scenario
bug-free compiler
bug-free middleware
Fault tolerant design
Designed for open-platform
(User may install new application)
Minimize system downtime for any unexpected event

You need to consider incomplete test coverage with the open-system.

Solid base + flexible app. management



Application rebooted but most users can not notice such incident.

Example : iOS5 secure application reboot log

```
Incident Identifier: 72FDDC4C-03EC-46B6-A707-134F96369D7F
CrashReporter Key:   b1ccfbf9e2a9bf5b44bf54fdf24a080c2dde875d
Hardware Model:      iPhone3,1
Process:         AngryBirdsFree [9903]
Path:            /var/mobile/Applications/3D2D8C5E-C7F5-47E7-8B82-135945E6BE36/AngryBirdsFree.app/AngryBirdsFree
Identifier:      AngryBirdsFree
Version:         ??? (???)
Code Type:       ARM (Native)
Parent Process:  launchd [1]

Date/Time:       2011-11-24 11:29:05.357 +0900
OS Version:      iPhone OS 5.0.1 (9A405)
Report Version:  104

Exception Type:  00000020
Exception Codes: 0x8badf00d
Highlighted Thread:  3

Application Specific Information:
AngryBirdsFree[9903] has active assertions beyond permitted time:
{(
    <SBProcessAssertion: 0x4a9020> identifier: Suspending process: AngryBirdsFree[9903] permittedBackgroundDuration:
      10.000000 reason: suspend owner pid:15 preventSuspend  preventThrottleDownCPU  preventThrottleDownUI
)}

Elapsed total CPU time (seconds): 5.720 (user 5.720, system 0.000), 57% CPU
Elapsed application CPU time (seconds): 0.082, 1% CPU
```

3rd topic for today’s talk

- High quality automotive produces + High quality Linux code = perfect match?
- “Bug free design” to “Fault tolerant design”
- How can you play with source code ?

You may want to add automotive demands

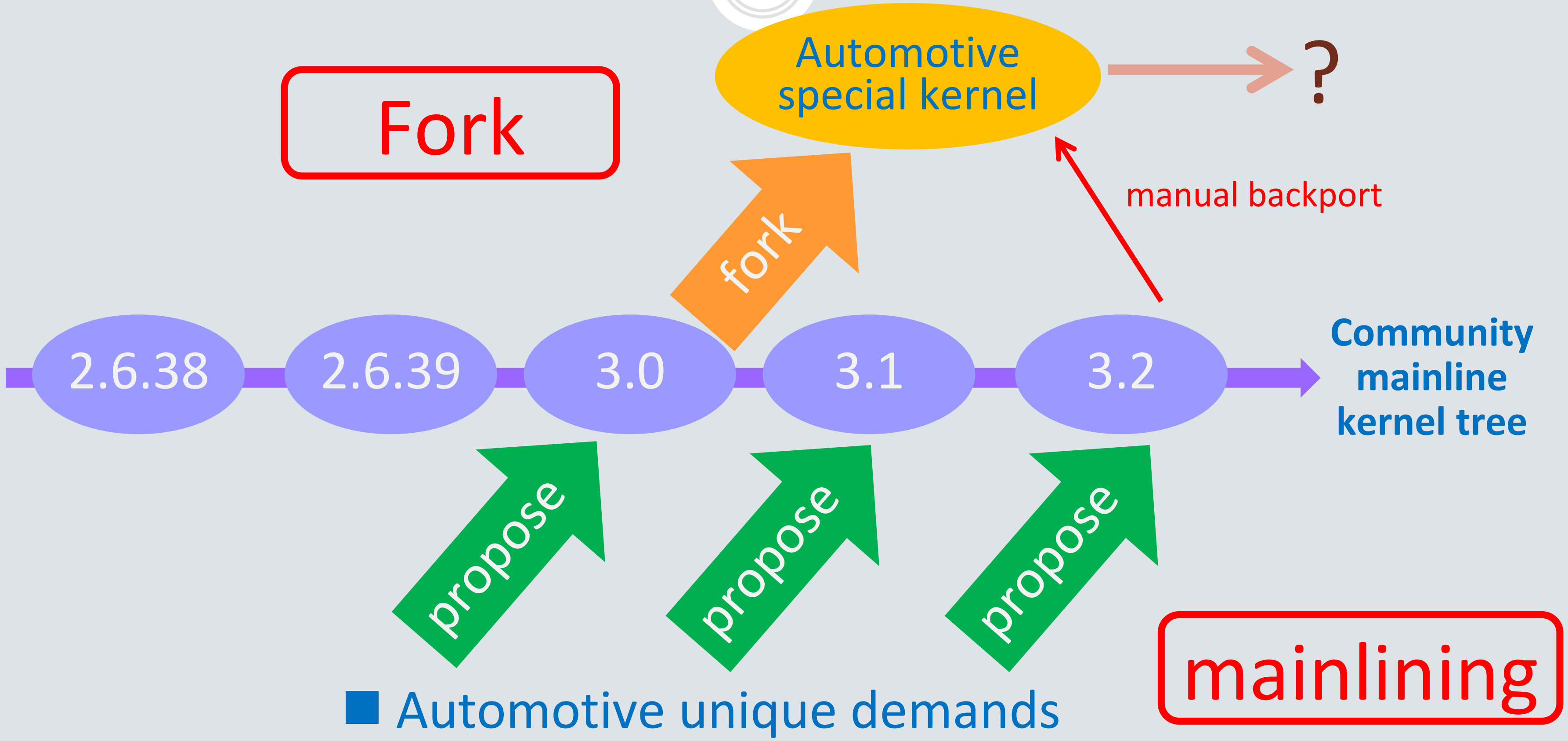
- High availability for non-stop operation
  - Silent reboot (for temporary power blackout recovery)
  - Seamless network hopping (3G <-> WiMAX <-> WiFi)
  - Smart data cache algorithm (for connection lost recovery)
- Long term support over 5 years
  - Security patch maintenance
  - Add new feature on top of stable kernel

There must be dedicated demands from automotive Linux adoption

How you can play with source code

- Linux kernel is open source licensed with GPL, therefore
  - You can modify kernel function whatever you want.
  - You can used modified code to your products.
  - Also you can re-distribute that code to anyone
- This may be very good for study, experimental trial, however
- I want to remind you that Linux kernel is
  - Highly complex and sensitive (to support abstraction)
  - Still adding new core feature incrementally (not freeze)
- So it is nice to know how other industry applied Linux first.

Fork or Mainlining



- Automotive unique demands
- Patch proposal & submission
- Open discussion on public ML

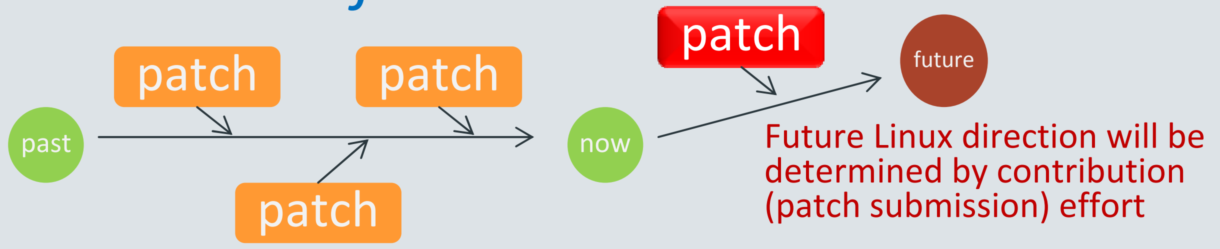
You can make own fork, but its maintenance cost will be very high

Who defines Linux spec ?

- Many industry people ask

  "Is there any roadmap, release plan and specification for the Linux kernel ? Who defines that ?"

- The answer is



patch
patch
patch
past
patch
now
future
Future Linux direction will be determined by contribution (patch submission) effort

  - There is no written roadmap, plan for Linux kernel.
  - Anyone who want to add/modify current kernel can send proposal with patch to each technology maintainer via ML.

- Current Linux include very wide-range adaptability from massive super-computer to tiny embedded products. All these works are done by patch-contribution effort.

Conclusion

- If automotive industry hopes to utilize Linux as open application platform, you may need to reconsider current business affiliation model, because Linux is too big and complicated to be cared by solely Tier1 development party.

- Quality is tied with requirement. And if we need to support open application platform that user can add application by themselves, it is getting very hard to proof full system level bug-free validation. So it may be necessary to accept new fault tolerant design to IVI system.

- If you want to modify Linux to fit automotive requirement, you can modify kernel code as it is licensed under GPL. But private fork would not be good idea and I really encourage automotive developers to participate in Linux community.