LTSI workshop 2012-4-17

- LTSI プロセス紹介 -

Hisao Munakata
CE WG, The Linux Foundation
Architecture Gr. co-chair

本日の話題

- LTSI コンセプト
- LTS/LTSI 関連用語定義
- LSTI バリュー
- LTSI リソース
- LTSI ルール
  - 開発プロセス定義
  - 評価プロセス定義
  - メンテナンスプロセス定義
- LTSI バグ対応
- ステータスアップデート
- 皆様へのお願い

LTSI コンセプト（1）

- “LTSI = community LTS kernel + 産業界の要求コード” で構成される Industry Tree.　Community tree とは別に管理される。産業界の利便性を考慮した独自運用ルールを制定可能

- 一定期間（数年規模）でコミュニティによって解決されたバグ修正、セキュリティ対策のメンテナンスリリースが行われる事が約束されている。（ Snapshot リリースではない ）

- Linux Foundation のメンバー費用で運営　LTSI 自体は公開資産であるが、LTSI のオペレーションルール、パッチ選定等は Linux Foundation の企業メンバーの意向を反映する。

- 公平性、透明性が必要であり、CEWG AG：Architecture Gr. がこれを担保する 審議機関 となる。

LTSI コンセプト（2）

- （デバイスドライバーを含む）カーネルにフォーカスした活動

  – 完全なディストリビューションを目指した活動ではなく
  いろいろな場面で広範に応用できる基盤となるカーネルを
  構築することを目指した活動である。
  ⇒ 自分たちなりに活用方法を考えて開発に適用するもの

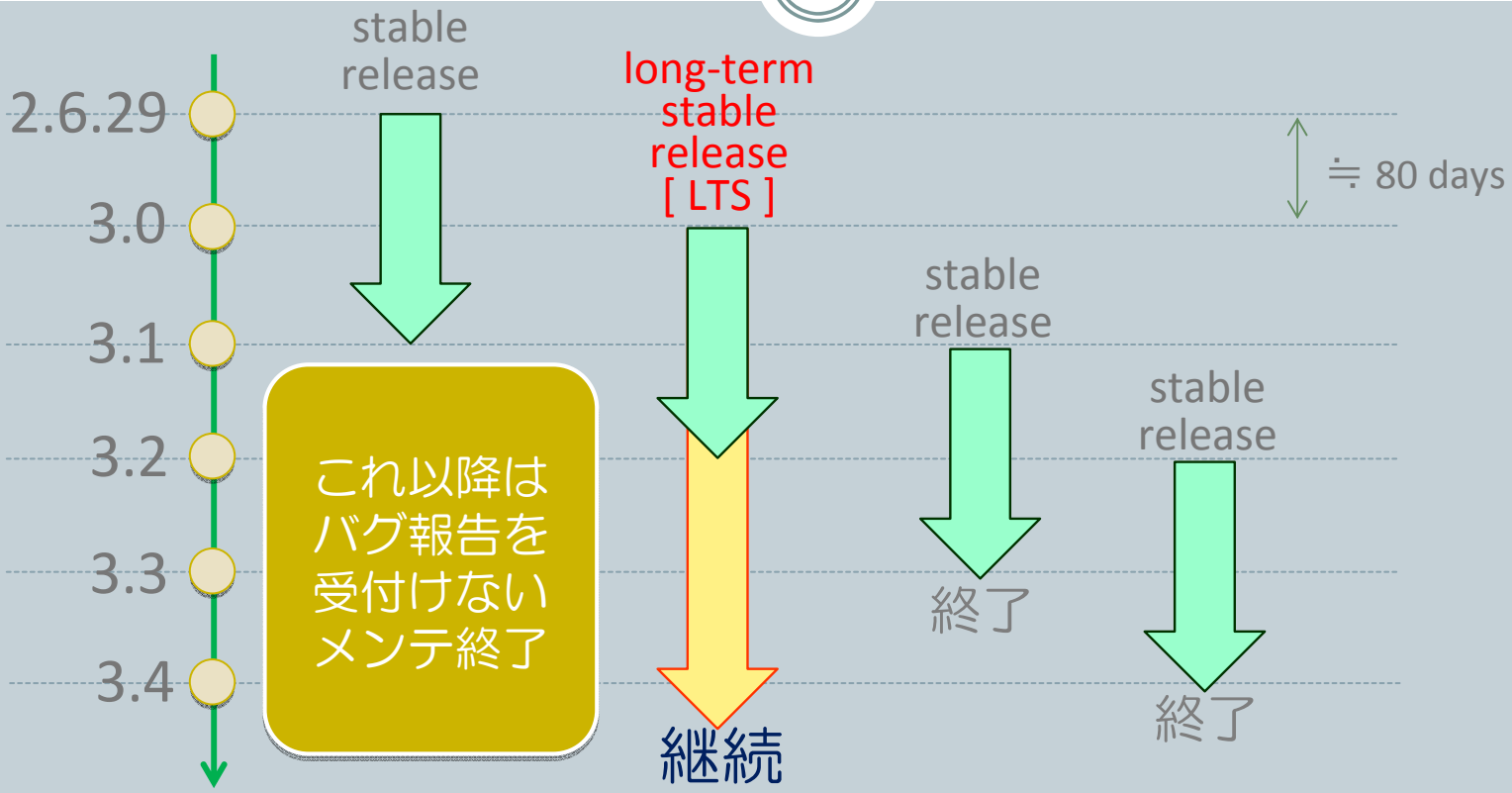
  – 既存の各種ディストリビューションのカーネルを LTSI 化
  する事で、双方にメリットが生まれることを期待している

    - 製品開発でのカーネルバージョンが揃ってくる事でカーネル
    部分のバグ修正などの知見が集約しやすくなる
    - Upstream に含まれていないコードを独自にディストロがチェリー
    ピックして独自の（そしてひとつひとつ微妙に違う）カーネルを
    提供する必要がなくなる。

LTS/LTSI 関連用語定義 (1/2)

- mainline
  - コミュニティーコード
  - 現時点でコミュニティがリリースした最新コード
  - 2つ先のバージョンが出るまでの期間限定でバグ修正を受付
- long-term stable [LTS] tree
  - コミュニティーコード
  - 2つ先のバージョンリリース以降も継続でバグ修正を受付
  - 開発途上のカーネルに採用された重要なバグ修正、セキュリティ対策のパッチを LTS バージョンにバックポート
  - rule : http://kernel.org/doc/Documentation/stable_kernel_rules.txt
  - LTS カーネルは常に最新のものに交換可能（更新を強く推奨）
    ⇒ 後方互換性確保のため、機能の拡張や追加は行わない

参考 ： リリース後のバグ修正取り込みの仕組み



community は期間限定でメンテナンス（バグ修正）に対応する

- 通常は 2つ先のバージョンのカーネルがリリースされるまで （リリース後 約5か月間）
- 特定バージョン （LTSバージョン）は一定期間メンテナンスが継続される

LTS/LTSI 関連用語定義 (2/2)

- LTSI tree
  - Linux Foundation のツリー
    (コミュニティによるメンテナンスの対象ではない)
  - コミュニティの LTS カーネルをベースに拡張する
    (バグ修正、セキュリティ対策はコミュニティ LTS に含まれる)
  - 産業界のニーズに対応したいくつかの拡張を行う
    但し、同じ LTSI バージョン内での後方互換性は確保し、常に最新のアップデートと交換可能とする。
    - デバイスドライバー、プラットフォームの追加
      ⇒ LTSI 3.0 では ARM device tree 対応は必須要件ではない
    - 最新カーネルで採用された新機能の取り込み
      (例えば 3.3 で採用された Android 拡張パッチ など)
    - Linux Foundation CEWG でファンドした組み込み向け拡張
    - SoCメーカー、製品メーカーの in-house コード（検証を実施）

THE LINUX FOUNDATION

Vanilla / LTS / LTSI

LTSI : long term support kernel

- Back-port newly added device / platform support from new kernel.
- Back-port new kernel feature to improve code functionality like PM.
  We may apply newly added (to 3.2 and later) Android code to LTSI3.0.
- SoC vender specific new code that are still in Linux-next can be added.
- Some embedded target kernel enhancement also can be added.

some feature back-port from newer kernel
!! SoC/manufacturer vendor code !!

Community long-term stable kernel

- Safe bug fix already verified and merged new kernel.
- Serious security fix already verified and merged new kernel.

fully confirmed bug-fix & security-fix

Upstream vanilla kernel

LTSI の提供価値（1）

- “原理的時間差問題” の解決
  - Linux BSP を入手した時点で既にそのカーネルの開発が終わっているという問題。製品開発の途上で重要なバグ修正やとても有益な拡張を行っても、それが反映できない
  - SoC ができあがった時点でプラットフォームのカーネルが既に決まっていて、デバイスやプラットフォーム対応のコードをマスターコードに登録できないので、仕方なく SoC ベンダーツリー（又はパッチ）を提供する必要がある

LTSI の提供価値（2）

- 開発コスト抑制
  - コード再利用性の拡大
    - 製品開発現場から提供されたパッチは、一般適用性が考慮されていないローカルフィックスである可能性が高いが、対策している問題は本質的である場合も多い。これらのコードをベースに LTSI マージ時点で汎用性のあるコードに修正し登録することを期待
      ⇒ パッチレビューと Upstreaming の支援を行う
  - コードインテグレーション工数の削減
    - 従来必ずしも全ての必要なコードが Upstream に登録されていなかったので、複数のツリーを統合していた
      ⇒ できるだけ多くのソースを LTSI に統合したい

LTSI の提供価値（3）

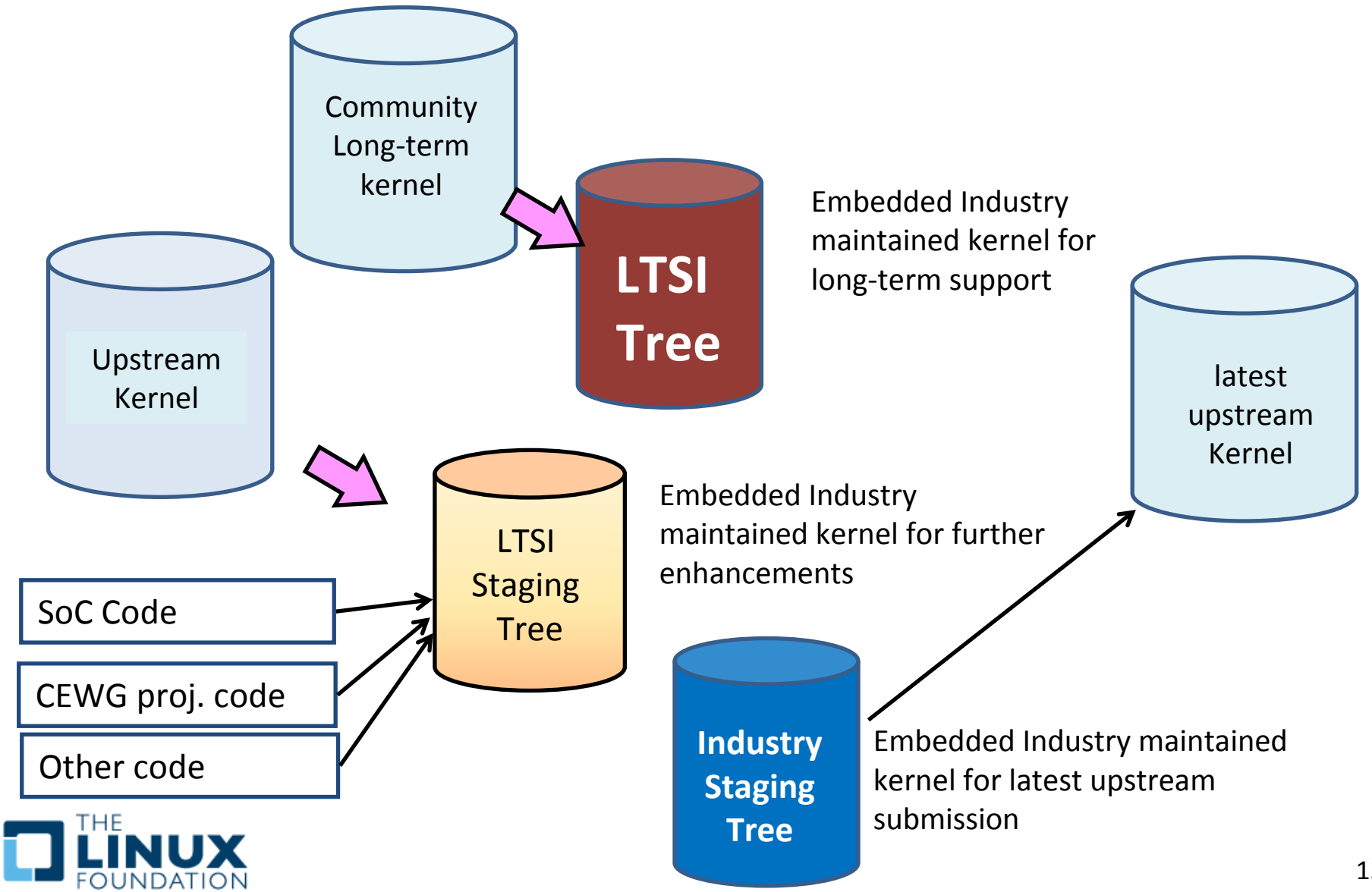
- Upstream との親和性、他の活動との整合性
  - Reviewed by Greg. K. H.
    - 従来から社内ディストロや特定用途向け民間？（商用でない）ディストリビューションを構築、メンテしている例もあり、期待された成果を上げている実例もある。

    - LTSI はこれらの特定用途向けディストロのカーネルと類似なものであるが、コミュニティーのキーマンである Greg がレビュー、マージを行う事から ① Upstream との整合性の良さ、② Yocto、Tizen など他の LF の活動との整合性、更に Genivi など他の活動やエンタープライズ向けディストロとの整合性などの可能性がある点が LTSI の大きな提供価値になる可能性がある。

LTSI リソース (1)

- 開発体制
  - LTSI chief maintainer = Greg Kroah-Hartman !
  - LTSI project manager = recruiting now
  - LTSI test & validation team = under considering
  - LTSI patch reviewer = LF/CEWG/AG → Chief maintainer
  - LTSI patch contributor = open to anyone
  - LTSI validation = SoC vendor, LTSI user

LTSI リソース (2)


Community
Long-term
kernel
LTSI
Tree
Embedded Industry
maintained kernel for
long-term support
Upstream
Kernel
latest
upstream
Kernel
LTSI
Staging
Tree
Embedded Industry
maintained kernel for further
enhancements
SoC Code
CEWG proj. code
Other code
Industry
Staging
Tree
Embedded Industry maintained
kernel for latest upstream
submission

LTSI リソース (3)

- ソースツリー
  - LTSI master tree（公開）→ リリース
  - LTSI staging tree（公開）→ パッチコレクション
  - Industry Staging tree (公開) → アップストリーム

- LTSI BTS
  - レビュー期間開始（6月初旬）までに開設予定

- メーリングリスト
  - LTSI メーリングリスト（公開）
  - LF/CEWG/AG メーリングリスト（メンバー専用）

- プロジェクトサイト
- ソーシャルメディア（ twitter ）

LTSI プロセス

- ① コレクションプロセス
  - パッチを集めてリリース候補を作るまでのプロセス
  - ベースとなる LTS バージョンの選定に依存する
- ② レビュープロセス
  - コレクションが完了して(＝マージウィンドウを閉じて)リリースに向けた検証を行う期間のプロセス
- ③ メンテナンスプロセス
  - リリースが行われたあと、バグ修正などメンテナンスを行うためのプロセス
- ④ 製品適用プロセス
  - LTSI を利用した Linux BSP を構築するプロセス

① LTSI コレクションプロセス（1）

- community LTS のバージョン決定プロセス
  - LTSI は community LTS をベースとするので、LTSI のバージョン選定は community の LTS による。
  - Community LTS は Greg が提案した新しいルールで運用
    - 毎年一つ LTS 候補にするバージョンを選定する
    - LTS に選定されたバージョンを 2年間メンテナンス (LTS のメンテナンス期間はユーザーの意向による)
  - LTS バージョンの選定ルールは明文化されていない
    - Greg は LTSI のチーフメンテナーだけでなく、広範にコミュニティ、インダストリー連携があり最大公約数を見つけるための努力をしてくれると期待

① LTSI コレクションプロセス（2）

- Upstream からのバックポート

  –既に最新の Upstream で採用済み（又は Linux-next にキューイングされていて採用が確実なもの）のコードについては原則として LTSI にインテグレーション可能。

- bug-fix、Security-fix コード（ LTS 部分）
  ⇒ 自動的にすべての LTS fix が LTSI master tree に取り込まれる

- feature backport

  –LTSI Staging tree 経由でマージする
  ⇒ 要求のあった機能だけがバックポートされる点に注意。
  !!黙っていては欲しいものは入らない !!

① LTSI コレクションプロセス（3）

- off-tree コードのインテグレーション
  - SoC のデバイスコード、プラットフォームコード
  - 製品開発途上で見つかったバグ修正、拡張
  - 組み込み機器に特化したカーネル拡張
    （ CEWG project funding の成果など）
  - その他のカーネル拡張コード

⇒ これらのコードは全て LTSI staging に一度プールして
全件レビュー、必要に応じて修正した上で LTSI の
reviewed-by sign をつけて LTSI マスター git に登録する
（ 但し LTSI3.0 の開発では マスター git を Staging tree として運用し、
LTSI 3.0 code freeze 時点で Staging tree をオープンする予定 ）

② LTSI レビュープロセス（1）

- 基本的に Upstream で採用済みのコード以外は全て LTSI Staging tree に一旦集約して、パッチ単位でレビューを実施
  - パッチ投稿時には必ず signed-off-by をつける
  - できるだけ Upstream のパッチ投稿ルールに準じる事が望ましいが、必ずしも Upstream 互換でないコードであってもレビューを受け付ける。 まずは相談してください。

  - フォーマッティング
  - 適切な抽象化
  - カーネル API の適切な利用

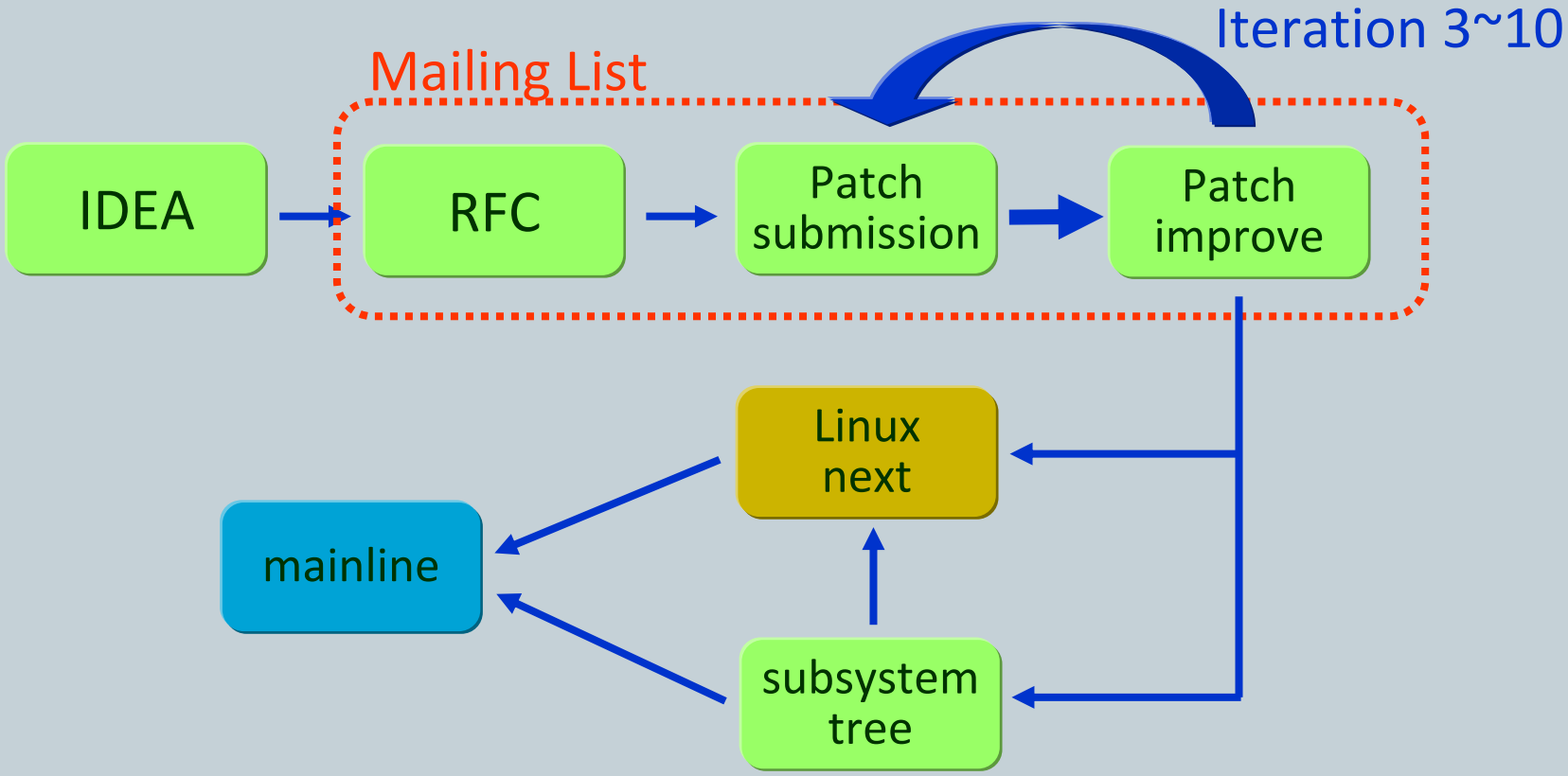
  } 修正依頼の可能性も

- 同時に最新カーネルへのパッチを作成してマスターコードの根本改善を支援する可能性がある。・・・・・ LTSI の真の狙いは従来パッチが投稿されていなかった業界からも価値あるフォードバックの反映

② LTSI レビュープロセス（2）

- LTSI は特定のターゲットハードウエア（ CPU アーキテクチャ、プラットフォーム ） を規定しない。
- Greg は x86 PC 環境で LTSI カーネルのビルド、起動ができることを確認するが、config の組み合わせの網羅性は足りない
- SoC ベンダー や Linux System Integrator は、公開されている LTSI master-git のコードをダウンロードして、自分たちの環境で実行、動作検証することができ、その結果を LTSI の検証結果として公開することができる（する事を期待している）
- ARM の Linaro などアーキテクチャに特化した検討を行う団体との連携（バグ情報の共有 等）も今後模索して議論中

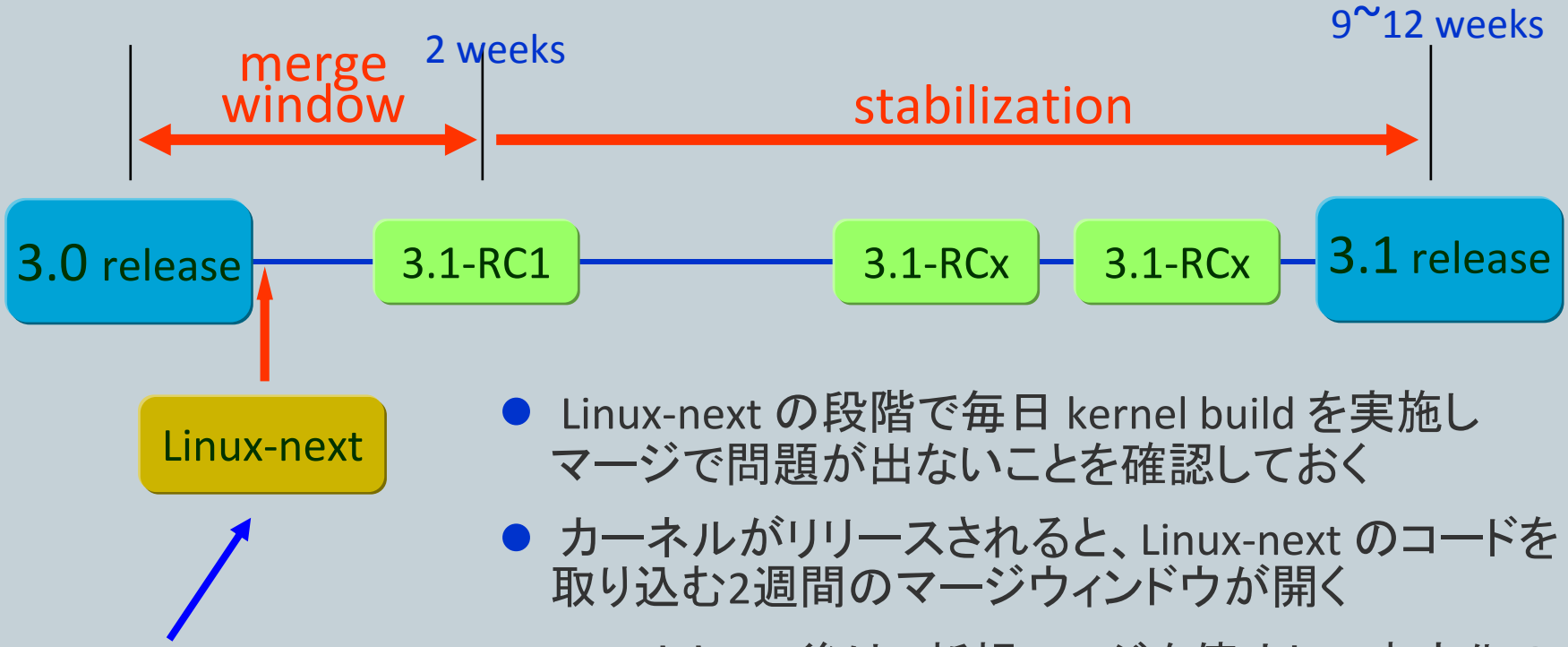
参考 ： パッチ投稿プロセスを理解する



ML上で提案を議論、 どんな経験者でも数回の書き直しを行って採用

Upstream kernel の

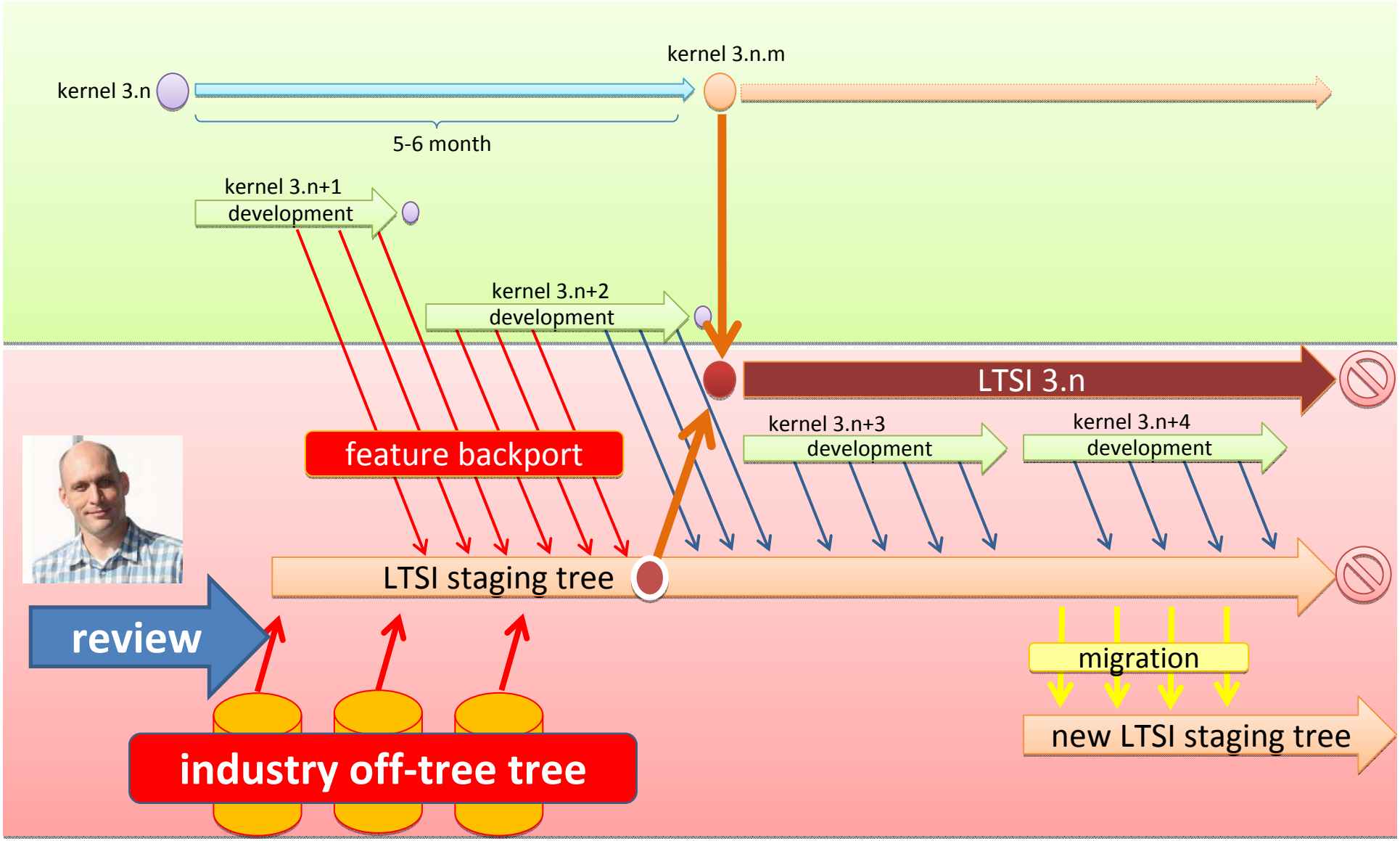
参考：コードレビュープロセスを理解する



- Linux-next の段階で毎日 kernel build を実施しマージで問題が出ないことを確認しておく
- カーネルがリリースされると、Linux-next のコードを取り込む2週間のマージウィンドウが開く
- RC1 リリース後は、新規マージを停止して安定化のためのシステム検証期間。通常は RC8~9 までいく

Linux-next に登録されてから最大22週間後にマージが完成する

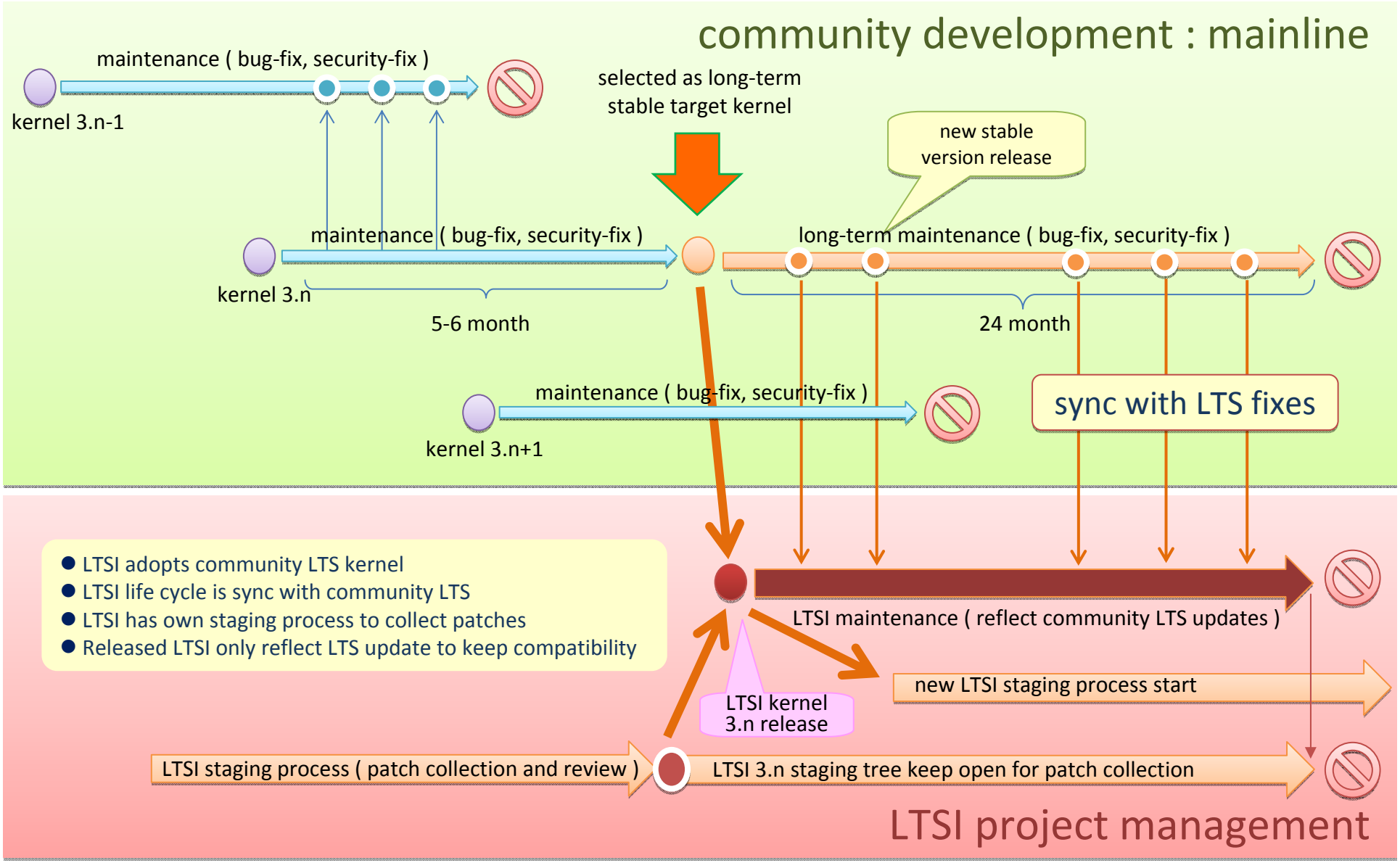
LTSI creation process



③ メンテナンスプロセス

- LTSI リリース後のメンテナンススキーム
  - Community LTS に逐次追加登録される バグ修正と セキュリティ対策は、その都度 LTSI に取り込まれる （ 自動的に全ての LTS 更新がLTSI に反映される ）
  - LTSI BTS に登録されたバグ修正コードは、レビュー が通った時点で LTSI マスターツリーに登録される （ 更新タイミングは LTS 修正反映と同期 ）
  - LTSI リリース後に Staging Tree に登録された新機能 新デバイス対応は、後方互換性を壊すリスクが無い ものだけを LTSI のアップデートに登録可能 ⇒ ここでは reviewed-by 3つ以上が必要等厳密な運用 を考える必要がある （ルール化を検討中）

LTSI development (after 2nd release)

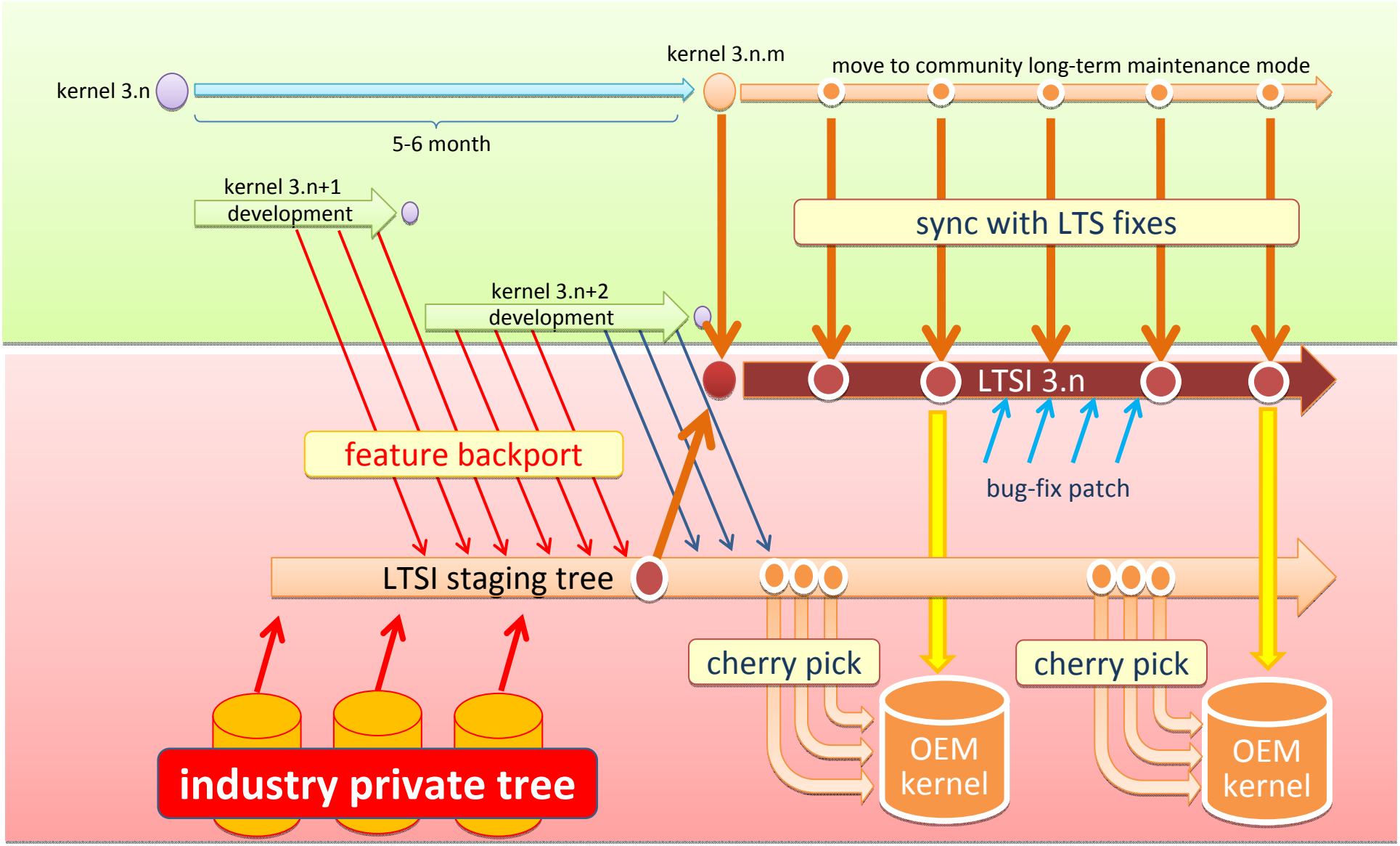


④ 製品開発時のプロセス

- 製品開発に適用する LTSI のバージョンを選定する
  - アプリケーションプラットフォームを選定した時に利用されているカーネルが LTSI でサポートされているか確認し、可能であれば LTSI カーネルを組み込んで利用

- LTSI で拡張されたパッチセット（機能、バグ修正 など）を確認
  - 特に必要なパッチがわかる場合には、そのパッチが既に LTSI カーネルに組み込まれているか確認する
  - 含まれていない場合、Staging Tree に該当パッチがプールされていないか更に確認する。（ LTSI リリース後には、後方互換性を崩すリスクが全くないもの以外は LTSI にマージせず Staging Tree にパッチとして集約する事がある）プールされたパッチをチェリーピックして独自 LTSI カーネルを構築して製品に利用することができる。

LTSI maintenance process



LTSI バグ対応

- Upstream のバグ修正、セキュリティ対策コードはLTS で適用されたパッチを自動的に LTSI に適用します

- LTSI 特有バグは LTSI BTS（Bug reporting & tracking system）［未開設］に登録してください。

- LTSI では将来的に独自のテストエンジニア、バグ修正エンジニアを雇う事を検討しますが、当面は発見者、当該 SoC コード提供者、その他ボランティアが協力して対策コードを作成して LTSI のパッチレビュープロセスで精査する事を想定します。各社の製品開発で発見、対策された修正を LTSI に集積する事でユーザー全体にメリットがある互恵関係を目指します。

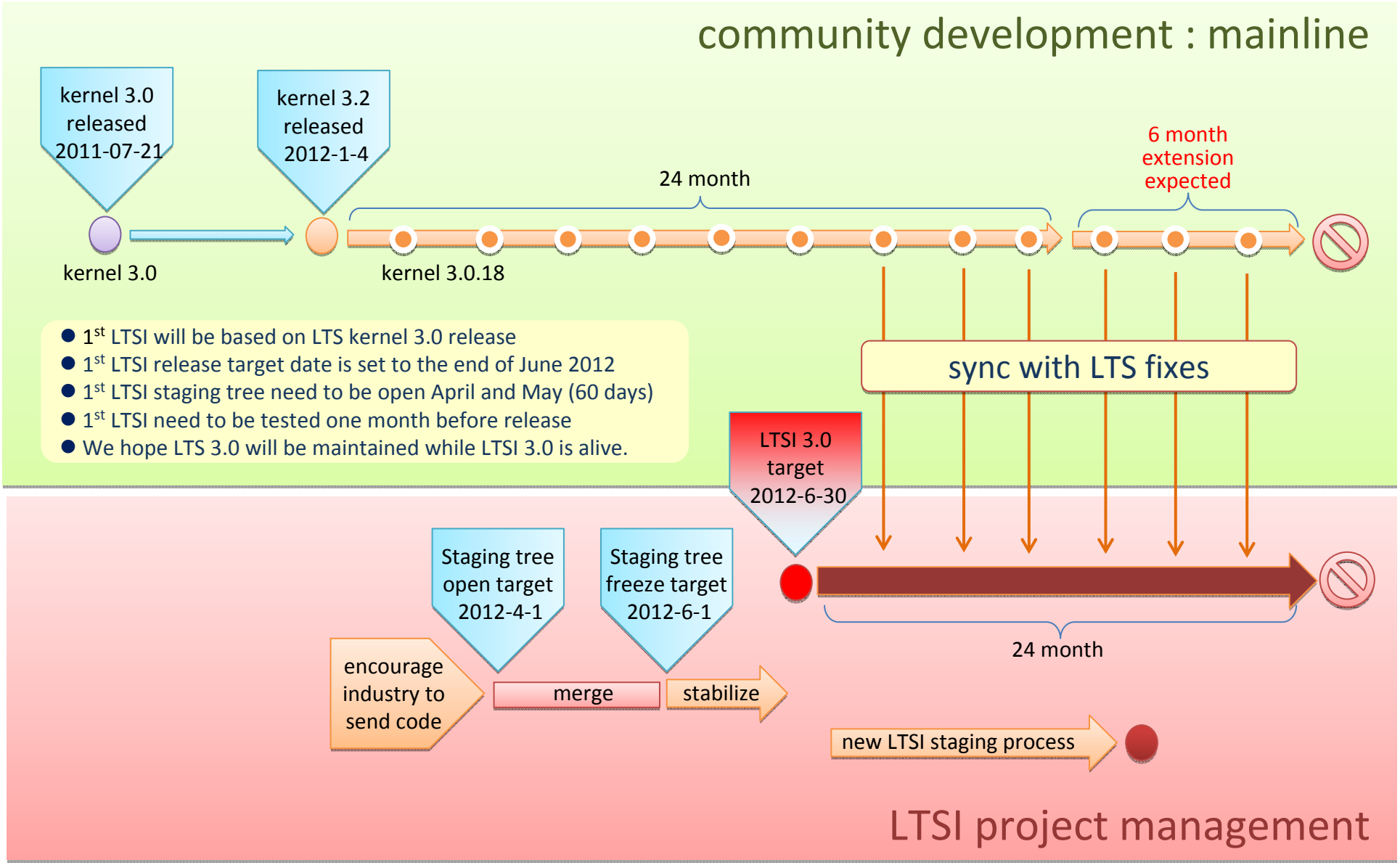
LTSI BTS 構想

- LTSI リリースまでに Bugzilla 等の BTS（ Bug reporting & tracking system）を公開開設予定。 ここでは
  - バグの報告
  - バグ対策のアサイメント（チケット管理）
  - バグ対策状況の閲覧
- メンバーとの議論の中で 各社が自社管理している BTS の issue ID と LTSI BTS issue ID のマッチング機構が必要との要望があり、実現方法を検討します。

ステータスアップデート

- ELCE2012（2011-10-26）にてプロジェクトキックオフ
- LTSI project web サイト公開
  - http://ltsi.linuxfoundation.org/
- LTSI git tree 公開
  - http://git.linuxfoundation.org/ltsi-kernel.git
- LTSI public ML 開設（メンバー登録制）
- Social Media 活用中（ twitter = @LinuxLTSI ）
- ELC2012 (2011-2-14) 併設で Partner meeting 実施
  - 約 20社が参加、当日ルネサスのコードをマージした
- 現在 SmartPhone ハンドセットベンダーを中心に LTSI チーフメンテナー の Greg さんと個別に投稿内容を相談中、まもなくgit にマージされる予定。各社準備進行中

LTSI 3.0 release plan



今やっていただきたい事

- LTSI 開発途上コードを DL して内容を確認
- マージ期間中（= 5月末まで）にパッチを出す
  - 最新カーネルからのバックポート
  - 現在 in-house で管理しているローカルパッチ
  - LTSI に登録したいデバイス、プラットフォーム
    （ LTSI 3.0 では DT : Device Tree 対応なしでも OK ）
- 評価期間（6月中）になったら
  - 自分の出したパッチの動作確認をする
  - SoC ベンダー中心で評価を実施し結果をレポート
  - 公開 ML 上での議論の活性化