การร่วมพัฒนาโครงการโอเพนซอร์ส

เทพพิทักษ์ การุญบุญญานันท์

thep@linux.thai.net

Thai Linux Working Group

การร่วมพัฒนาโครงการโอเพนซอร์ส

- โอเพนซอร์สทำงานได้อย่างไร
- องค์ประกอบของประชาคม
- เครื่องมือสื่อสาร
- ข้อปฏิบัติ
- การจัดการโครงการ

โอเพนซอร์สทำงานได้อย่างไร

- ผู้ใช้สามารถศึกษาการทำงานของโปรแกรมโดยสะดวก

  $\longrightarrow$ ทางลัดของการเกิดผู้เชี่ยวชาญ
- ผู้ใช้สามารถดัดแปลงโปรแกรมตามความต้องการ

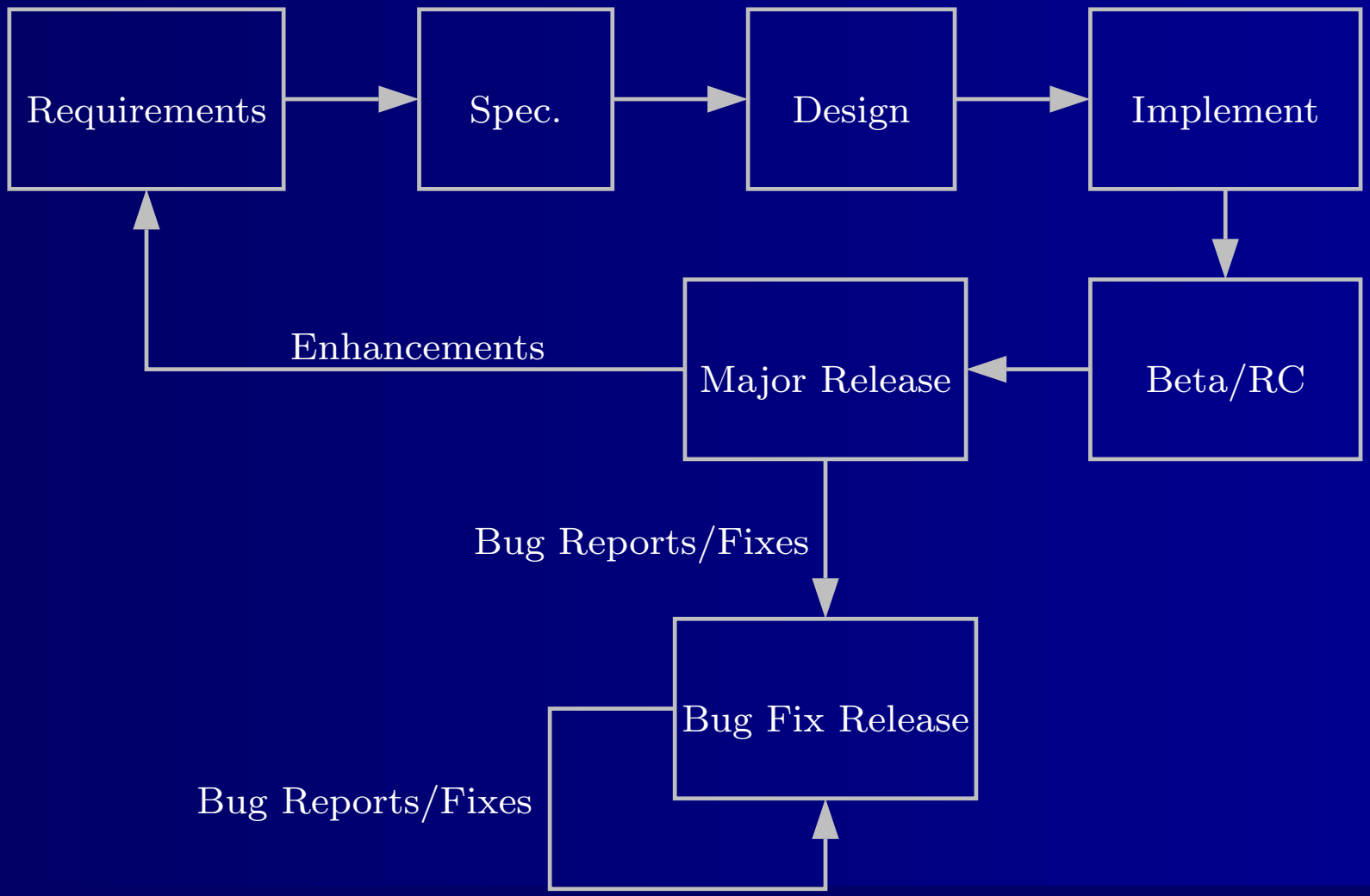
  $\longrightarrow$ ทางลัดของการแก้ปัญหาด้วยตนเอง
- ผู้ใช้สามารถแบ่งปันการปรับปรุง

  $\longrightarrow$ ทางลัดของการร่วมพัฒนา

$\downarrow$

ผู้ใช้คือผู้ร่วมพัฒนาโปรแกรม!

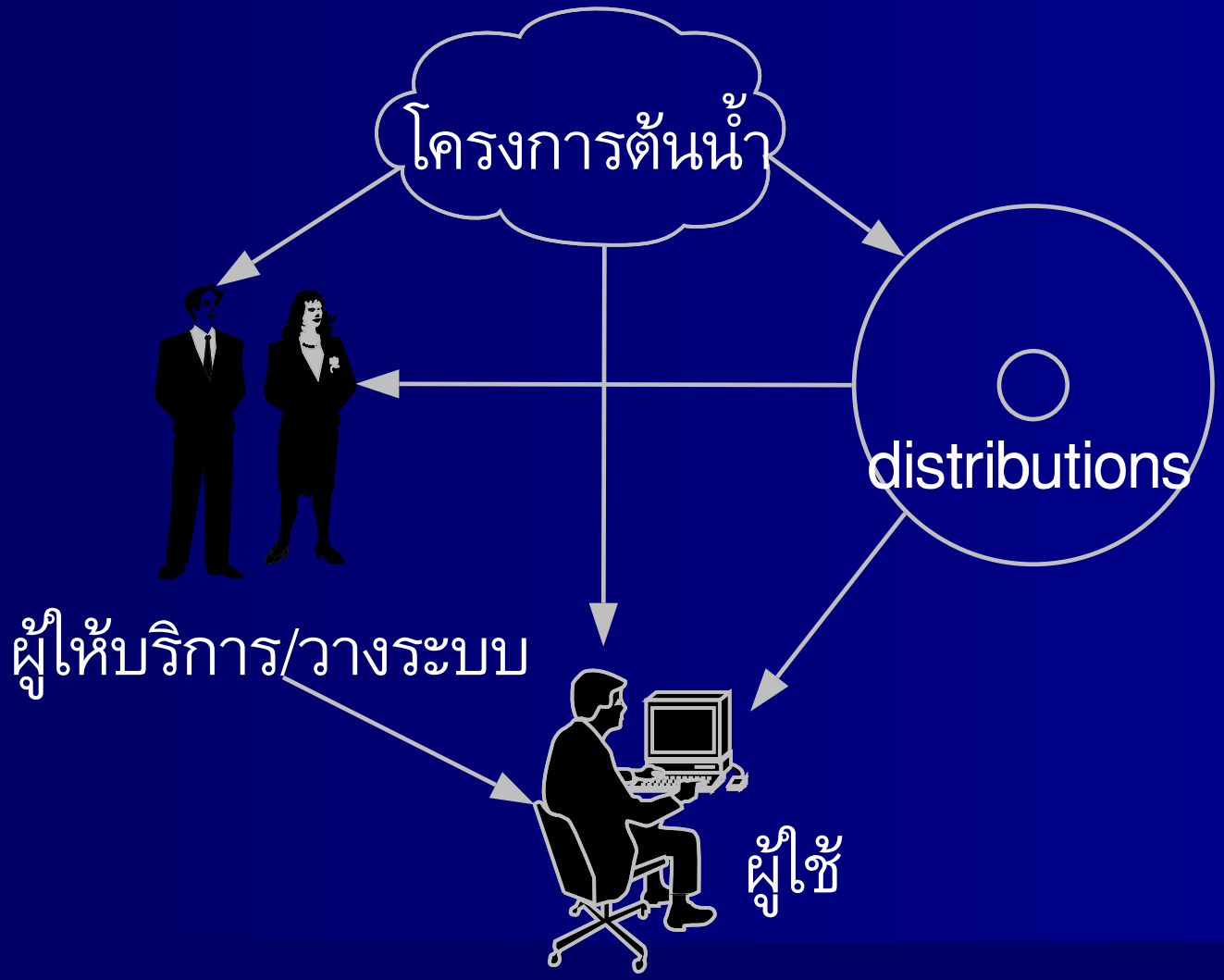
โอเพนซอร์สทำงานได้อย่างไร



องค์ประกอบของประชาคม

๑. โครงการต้นน้ำ (upstream projects)
๒. ผู้รวบรวมแจกจ่าย (distributions)
๓. ผู้ให้บริการสนับสนุนและวางระบบ (supporters & solution providers)
๔. ผู้ใช้ (users)

องค์ประกอบของประชาคม



องค์ประกอบของประชาคม

- ในห่วงโซ่การบริโภค จะมีการป้อนกลับเสมอ
- การบริโภคคือวิถีชีวิต การป้อนกลับคือการเติบโต
- ตัวอย่างการป้อนกลับ $\longrightarrow$ โครงการต้นน้ำ
  - รายงานข้อผิดพลาดของโปรแกรม
  - patch
  - ข้อเสนอแนะปรับปรุง
  - รายงานข้อความที่แปล/สะกดผิด
  - ช่วยเขียนเอกสารประกอบ
  - ช่วยแปลข้อความ

องค์ประกอบของประชาคม

- ตัวอย่างการป้อนกลับ → distributions
  - รายงานข้อผิดพลาดของการติดตั้ง
  - ข้อเสนอแนะปรับปรุง
  - แจ้งผู้ดูแลเมื่อโครงการต้นน้ำออกโปรแกรมรุ่นใหม่
  - ช่วยเขียนเอกสารประกอบ
  - (เฉพาะบาง distro) สมัครเป็นผู้ดูแลแพกเกจ
- ตัวอย่างการป้อนกลับ → ผู้ใช้ด้วยกัน
  - ตอบคำถามเรื่องการใช้งาน
  - เขียนเอกสาร tutorial, HOWTO

เครื่องมือสื่อสาร

- mailing list
- webboard
- chat (IRC)
- bugzilla

ข้อปฏิบัติ

- การเริ่มโครงการ
- การร่วมประชาคม
- การร่วมพัฒนา

ข้อปฏิบัติ — การเริ่มโครงการ

- สำรวจงานที่มีอยู่ก่อนเริ่มโครงการ
  - ใช้ search engine, freshmeat, sourceforge หรือถามใน mailing list
  - ถ้ามีงานเดิมอยู่แล้ว $\longrightarrow$ พยายามร่วมมือพัฒนา
  - ถ้ามีงานเดิมที่ใช้ประกอบได้ $\longrightarrow$ พยายามใช้
  - อย่าลืมว่าโอเพนซอร์สได้ทะลายกำแพงเรื่อง license แล้ว
  - ไม่มีใครทำมาก่อนจริง ๆ $\longrightarrow$ เริ่มโครงการเอง

ข้อปฏิบัติ — การร่วมประชาคม

- พยายามช่วยตัวเอง
- ใช้ mailing list, เลี่ยงการ mail ตรง
- อย่าบ่น อย่าชี้นิ้ว (ทุกคนคืออาสาสมัคร)
- อย่าพยายามพูดในสิ่งที่ตนไม่รู้
- เคารพความเห็นของผู้ดูแลโครงการ
- ติดตามข่าวสาร (จะได้ไม่เบื่อ)
- เตรียมตัวโดนด่า (มันคืออินเทอร์เน็ต)

ข้อปฏิบัติ — การร่วมพัฒนา

- รายงาน bug ⟶ ต้องชัดเจน
  - พยายามใช้ bug tracking system ดีกว่า mailing list
  - ค้นหา bug เก่าก่อนรายงานซ้ำ
  - ระบุอาการ, เงื่อนไขที่ทำให้เกิด bug
  - ระบุสิ่งที่คาดหวัง เทียบกับสิ่งที่เกิด
  - screenshot, ข้อมูลสำหรับทดสอบ (ถ้ามี)
- ช่วยตรวจหาสาเหตุ

ข้อปฏิบัติ — การร่วมพัฒนา

- เสนอ patch สำหรับแก้ bug
  - พยายามแก้แต่น้อย ทำ patch ให้เล็ก
  - ระวังเรื่อง coding style
- patch review (มักกระทำโดยผู้ดูแลโครงการ)
  - ทดสอบ patch
  - เสนอแนะการปรับปรุง patch

ข้อปฏิบัติ — การร่วมพัฒนา

- ตัดสินใจ (มักกระทำโดยผู้ดูแลโครงการ)
  - ตกลงใช้ patch
  - ให้ QA patch ก่อนใช้
  - เป็น bug ที่เคยพบมาแล้ว
  - เปลี่ยน priority ของ bug
  - ไม่ใช่ bug

การจัดการโครงการ

- แนวคิดของโครงการ
- license
- design
- versioning
- การสร้างประชาคม
- การติดตาม bug

การจัดการโครงการ — แนวคิดของโครงการ

- ทำไมถึงทำ?
- platform เป้าหมาย
- ลักษณะการใช้งาน
- สถาปัตยกรรม
- ภาษาที่ใช้พัฒนา

การจัดการโครงการ — License

- ควรตัดสินใจในขั้นต้น ๆ ของการออกแบบ หรือก่อนประกาศต่อสาธารณะ
- พิจารณาตามกลุ่มเป้าหมาย
  - เปิดให้ใช้ในเชิงพาณิชย์เต็มที่ $\longrightarrow$ MIT, BSDL
  - ให้ใช้ร่วมกับโปรแกรมเชิงพาณิชย์ได้ แต่ปกป้องเสรีภาพในตัวซอฟต์แวร์เอง $\longrightarrow$ LGPL
  - ปกป้องเสรีภาพของตัวซอฟต์แวร์เต็มที่ $\longrightarrow$ GPL
  - เงื่อนไขอื่น ๆ $\longrightarrow$ ศึกษาจาก www.opensource.org

การจัดการโครงการ — License

- *ไม่แนะนำ*ให้สร้าง license ใหม่
  - เลี่ยงปัญหาความไม่เข้ากันของ license
  - เพิ่มความสะดวกในการใช้งาน

การจัดการโครงการ — Design

- ความสำคัญของ design
  - ความเข้าใจที่ตรงกันระหว่างผู้ร่วมพัฒนา
  - ความเข้ากันได้กับโปรแกรมอื่น
  - เหตุผลด้านวิศวกรรมซอฟต์แวร์ตามปกติ

การจัดการโครงการ — Design

- design อะไรบ้าง?
  - หลักการ ทฤษฎี
  - โครงสร้าง ส่วนประกอบ
  - สถาปัตยกรรมของการเชื่อมต่อ
  - API
  - โครงสร้างไดเรกทอรีของ source code
  - coding style

การจัดการโครงการ — Versioning

- ระบบการนับเวอร์ชันที่นิยม: *major.minor.micro*
  - *major* — การเปลี่ยนสถาปัตยกรรมขนานใหญ่
  - *minor* — การเพิ่ม feature
  - *micro* — การแก้ bug, maintenance

การจัดการโครงการ — Versioning

- การแบ่ง development/stable release
  - development — เพื่อให้นักพัฒนาทดสอบแก้ไข
  - stable — เพื่อให้ผู้ใช้ใช้
  - Linux kernel, GNOME:
    - *minor* คี่ = development release
    - *minor* คู่ = stable release

การจัดการโครงการ — การสร้างประชาคม

- เครื่องมือหลัก: web page
  - รายละเอียดโครงการ
  - ข่าวคราวของโครงการ
  - download
  - FAQ
  - เอกสารประกอบ
  - contact address
  - ช่องทางการร่วมพัฒนา, รายงาน bug

การจัดการโครงการ — การสร้างประชาคม

- ช่องทางการร่วมพัฒนา:
  - mailing list สำหรับอภิปราย
  - CVS repository สำหรับเก็บซอร์สกลาง
  - bugzilla สำหรับติดตาม bug
  - IRC สำหรับการอภิปรายแบบ on-line
  - Wiki สำหรับการร่วมเขียนเอกสารผ่าน web

การจัดการโครงการ — การติดตาม Bug

- การติดตาม bug อย่างง่าย: mailing list
  - ข้อเสีย: ใช้ความจำมนุษย์ล้วน ๆ

    $\longrightarrow$ ไม่สามารถติดตาม bug จำนวนมากได้หมด

การจัดการโครงการ — การติดตาม Bug

- bugzilla (จากโครงการ mozilla):
  - จัดการ bug ของซอฟต์แวร์หลายตัวได้
  - จัดการ bug แยกเป็นเรื่องย่อยได้
  - จัด priority ของ bug ได้
  - แก้ bug จบเป็นข้อ ๆ
  - ค้นหา bug เก่าได้

การจัดการโครงการ — การติดตาม Bug

- Debian: reportbug
  - bug database โดยใช้ระบบ mail
  - คำสั่ง reportbug
    - ตรวจสอบซอฟต์แวร์รุ่นใหม่ก่อนรายงาน
    - ค้นหา bug เก่าก่อนรายงาน
    - รายงาน bug พร้อมแนบข้อมูลแพกเกจโดยอัตโนมัติ

การจัดการโครงการ — การติดตาม Bug

- GNOME: bug-buddy
  - โปรแกรมช่วยรายงาน bug เมื่อโปรแกรม crash
  - อ่าน core file และแนบข้อมูล stack โดยอัตโนมัติ
  - ช่วยส่งข้อมูลเข้า GNOME bugzilla

ขอบคุณครับ

Happy Hacking!!