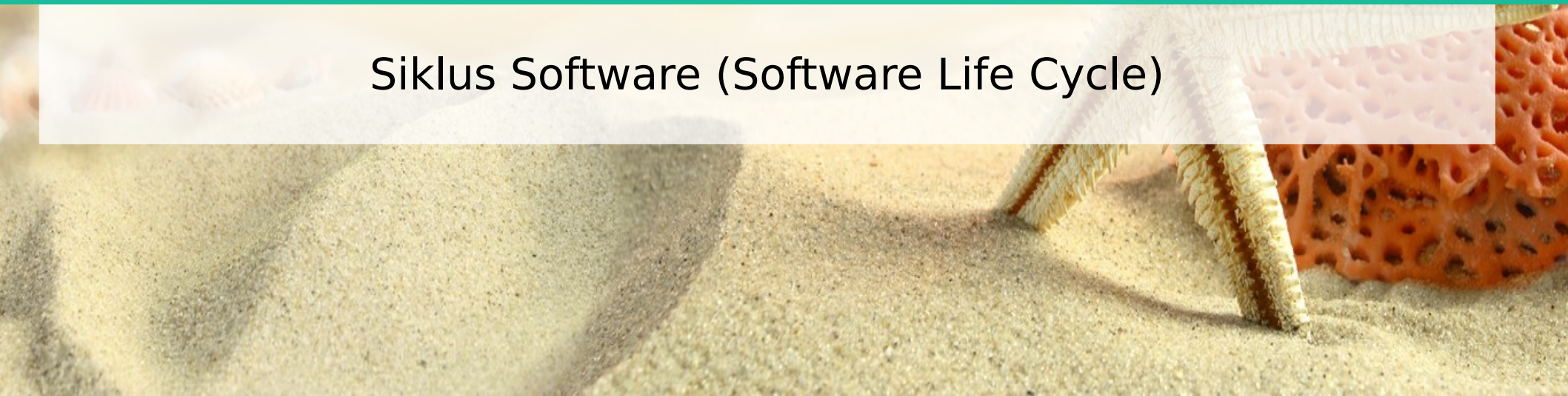
Rekayasa Perangkat Lunak

Siklus Software (Software Life Cycle)



Siklus Hidup Software

Siklus hidup software

- **Setiap program, berapapun ukurannya, punya siklus hidup:**
  1. Konsep
  2. Spesifikasi/Eksplorasi/Pemodelan
  3. Desain
  4. Penulisan Kode dan Debugging
  5. Testing
  6. Rilis
  7. Pemeliharaan
  8. Pensiun

Siklus Hidup

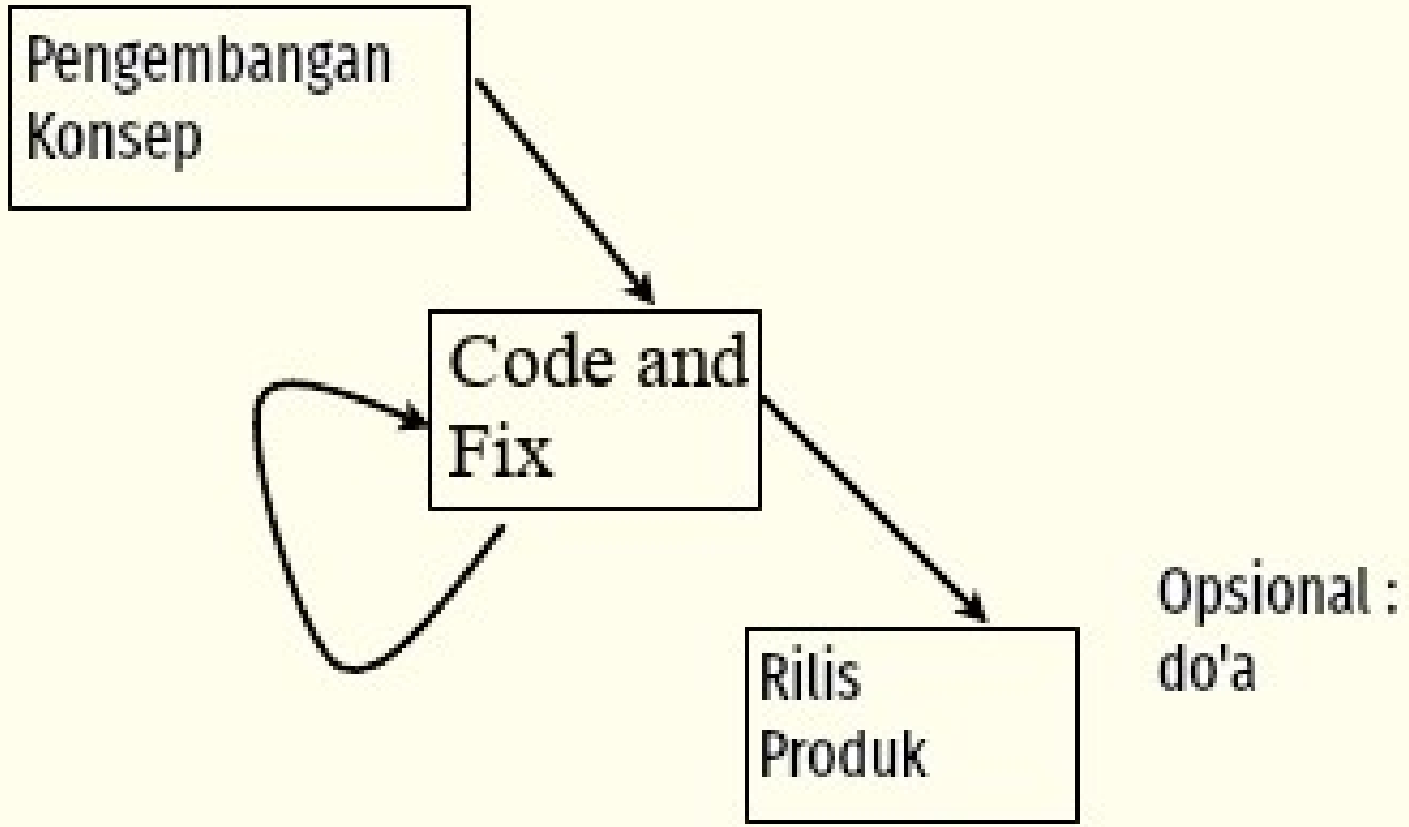
- Setiap fase bisa saling tumpang tindih
- Bisa menyatu
- Untuk program yang kecil, anda mungkin tak menyadari telah melalui satu atau beberapa fase
- Akan tetapi selalu terjadi dalam penulisan program

Model Siklus Hidup

- **Setiap model siklus adalah variasi dari 2 tipe fundamental:**
  - Melakukan seluruh siklus , step 2 – 7 dan (kadang) mulai lagi dari awal
  - Melakukan sebagian, biasanya step 3 – 5, membuat prototip, dan kemudian membuat lagi sebagai hal baru atau mengklarifikasi spesifikasi kebutuhan (requirement) yang dibuat

1. **Konsep**
2. **Spesifikasi/Eksplorasi/ Pemodelan**
3. **Desain**
4. **Penulisan Kode dan Debugging**
5. **Testing**
6. **Rilis**
7. **Pemeliharaan**
8. **Pensiun**

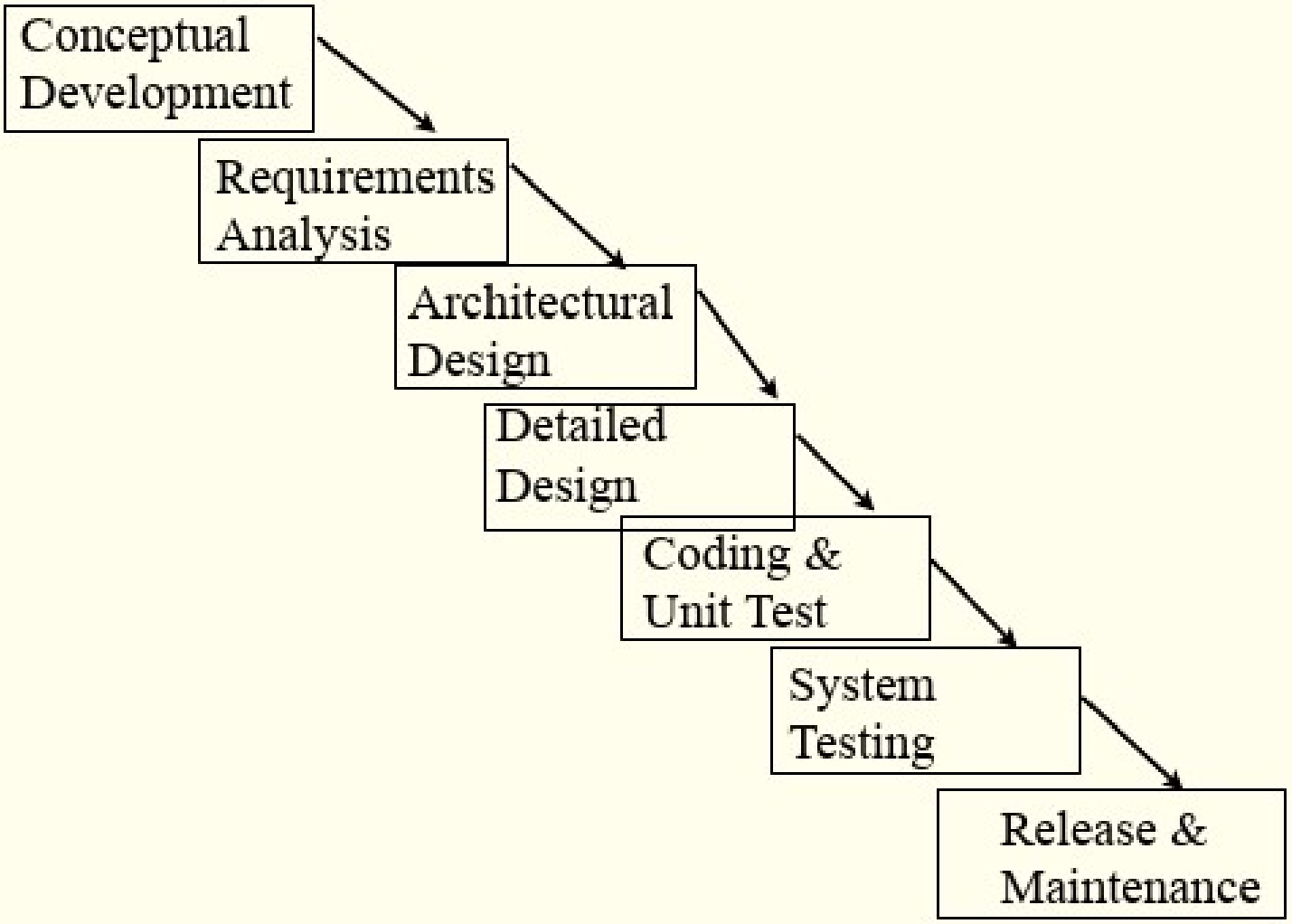
Model “Koding dan Perbaiki“



Model Koding dan Perbaiki

- Lebih banyak digunakan daripada manajemen proyek
- Tidak ada spsifikasi kebutuhan secara formal, dokumentasi, QA, testing dll
- Bekerja baik untuk pekerjaan yang cepat dan disposable (misalnya untuk bukti konsep) – tidak ada maintenance
- Cocok untuk proyek personal
- SANGAT BERBAHAYA untuk pengembangan software selain hal di atas

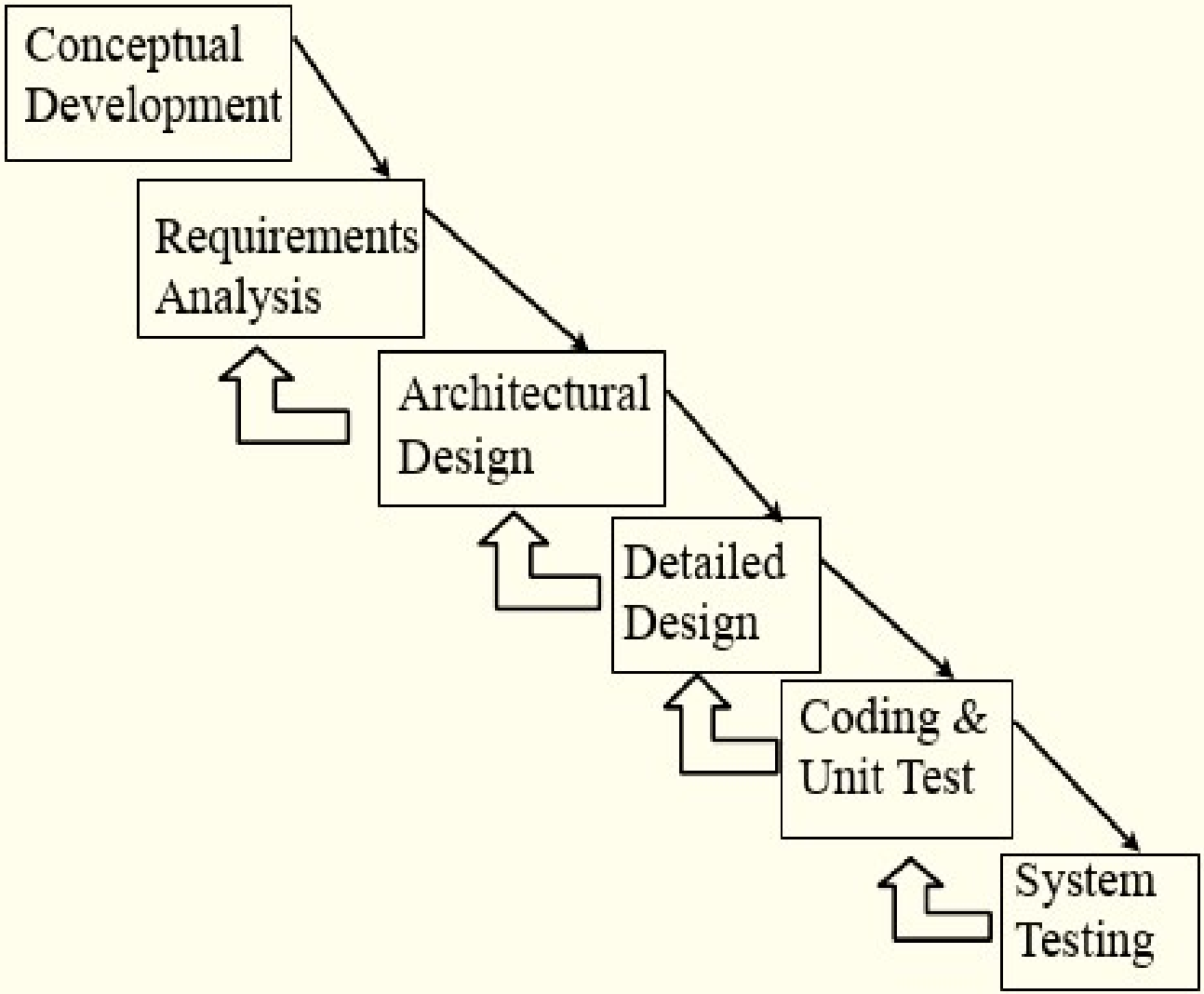
Model Waterfall klasik



Model Waterfall klasik

- **Baik untuk:**
  - Spesifikasi Kebutuhan sudah diketahui DAN teknologi juga sudah diketahui
  - Staf yang lemah dan tidak berpengalaman
  - Spesifikasi kebutuhan yang berkualitas tinggi (ini harus …)
- **Problem:**
  - Sangat kaku
  - Hampir tidak mungkin untuk bisa menspesifikasikan kebutuhan dari awal secara keseluruhan
  - Sedikit petunjuk untuk proses yang sedang berjlan
  - Tidak ada data ynag bisa digunakan selama progres hingga fase iimplementasi
- **Pada kenyataannya tidak bisa digunakan secara nyata**

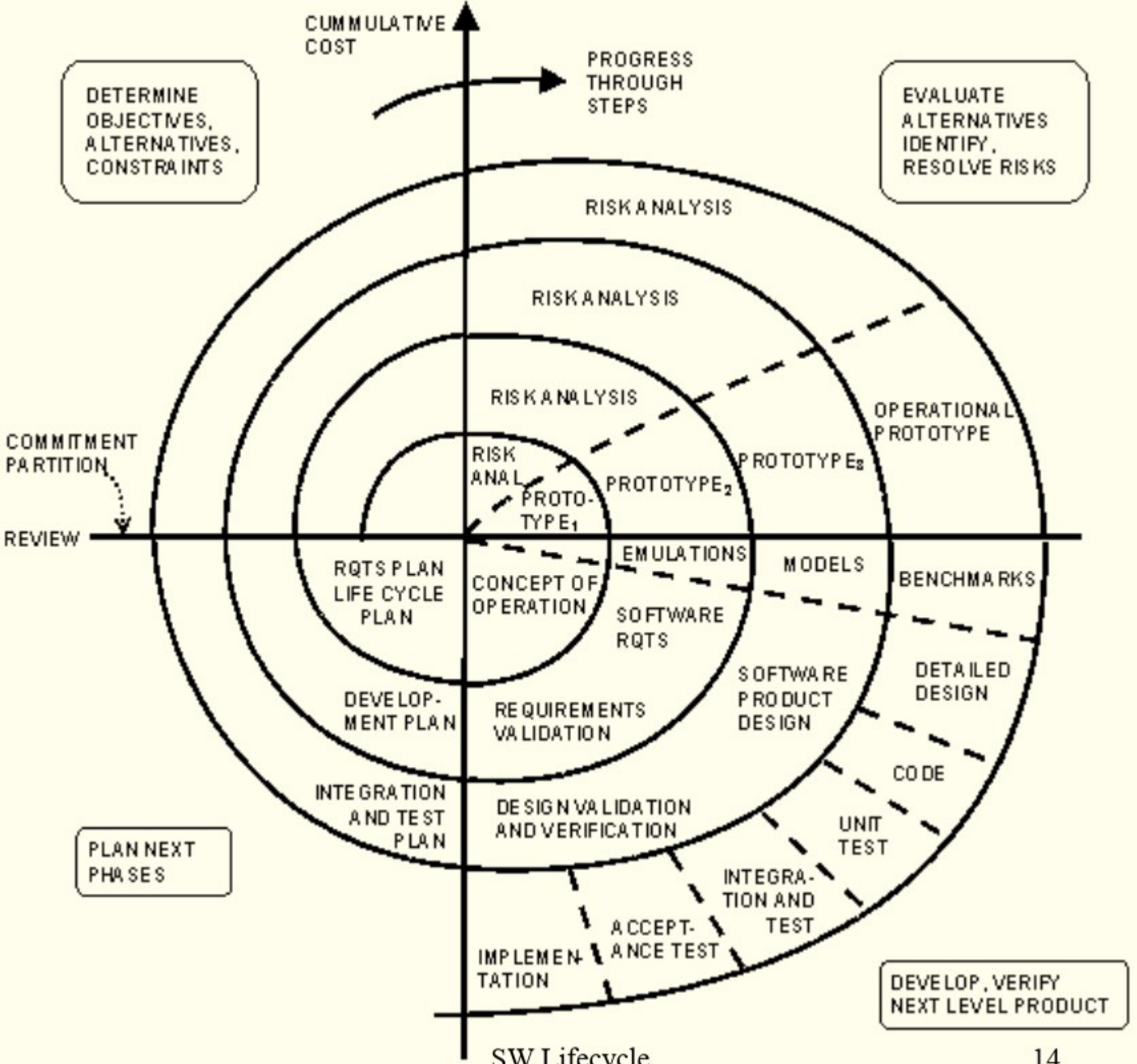
Waterfall dengan Feedback



- Tetap kaku, walaupun memungkinkan untuk kembali ke tahap sebelumnya jika ada informasi baru.
- Kelemaahan utama: jadwal sangat tidak bisa diduga...proyek bisa molor berkepanjangan

Model Siklus Spiral (Spiral Life Cycle Model)

- Merupakan model inkremental yang pertama
- Dikembangkan oleh Boehm 1988
- Fokus pada antisipasi risiko mayor pada proyek (requirements, arsitektur, dll)
- Setiap "proyek mini" mengantisipasi satu atau beberapa area risiko, kemudian berlanjut pada proyek mini berikutnya


CUMMULATIVE COST
PROGRESS THROUGH STEPS
DETERMINE OBJECTIVES, ALTERNATIVES, CONSTRAINTS
EVALUATE ALTERNATIVES IDENTIFY, RESOLVE RISKS
RISK ANALYSIS
RISK ANALYSIS
RISK ANALYSIS
COMMITMENT PARTITION
REVIEW
RISK ANAL
PROTO-TYPE1
PROTOTYPE2
PROTOTYPE3
OPERATIONAL PROTOTYPE
RQTS PLAN LIFE CYCLE PLAN
CONCEPT OF OPERATION
EMULATIONS
SOFTWARE RQTS
MODELS
BENCHMARKS
DEVELOP-MENT PLAN
REQUIREMENTS VALIDATION
SOFTWARE PRODUCT DESIGN
DETAILED DESIGN
CODE
INTEGRATION AND TEST PLAN
DESIGN VALIDATION AND VERIFICATION
UNIT TEST
INTEGRA-TION AND TEST
ACCEPT-ANCE TEST
IMPLEMEN-TATION
PLAN NEXT PHASES
DEVELOP, VERIFY NEXT LEVEL PRODUCT

Model Spiral

- **Biasanya dikombinasikan dengan model siklus yang lain dalam setiap proyek mini**
- **Keunggulan:**
  - Menangani risiko dengan baik
- **Kelemahan:**
  - Sangat komplek
  - Sulit untuk mendefinisikan dan menyelesaikan banyak proyek mini'
  - Sulit untuk fokus pada tujuan proyek keseluruhan

Pengembangan secara Iterativ

- Praktek terbaik adalah dengan iterate dan deliver secara inkremntal
- Memperlakukan setiap iterasi sebagai proyek mini yang komplet termasuk rquirements, desain, coding, integration, testing, dan delivery iternal. Pada deadline itersi, sistem yang teruji dan terintegrasi secara penuh di sajikan pada stakeholders internal, lalu catat feedback terhadap hasilnya, lalu masukkan dalam iterasi selanjutnya.

Satu aturan dasar untuk model iterativ (Tom de Marco):

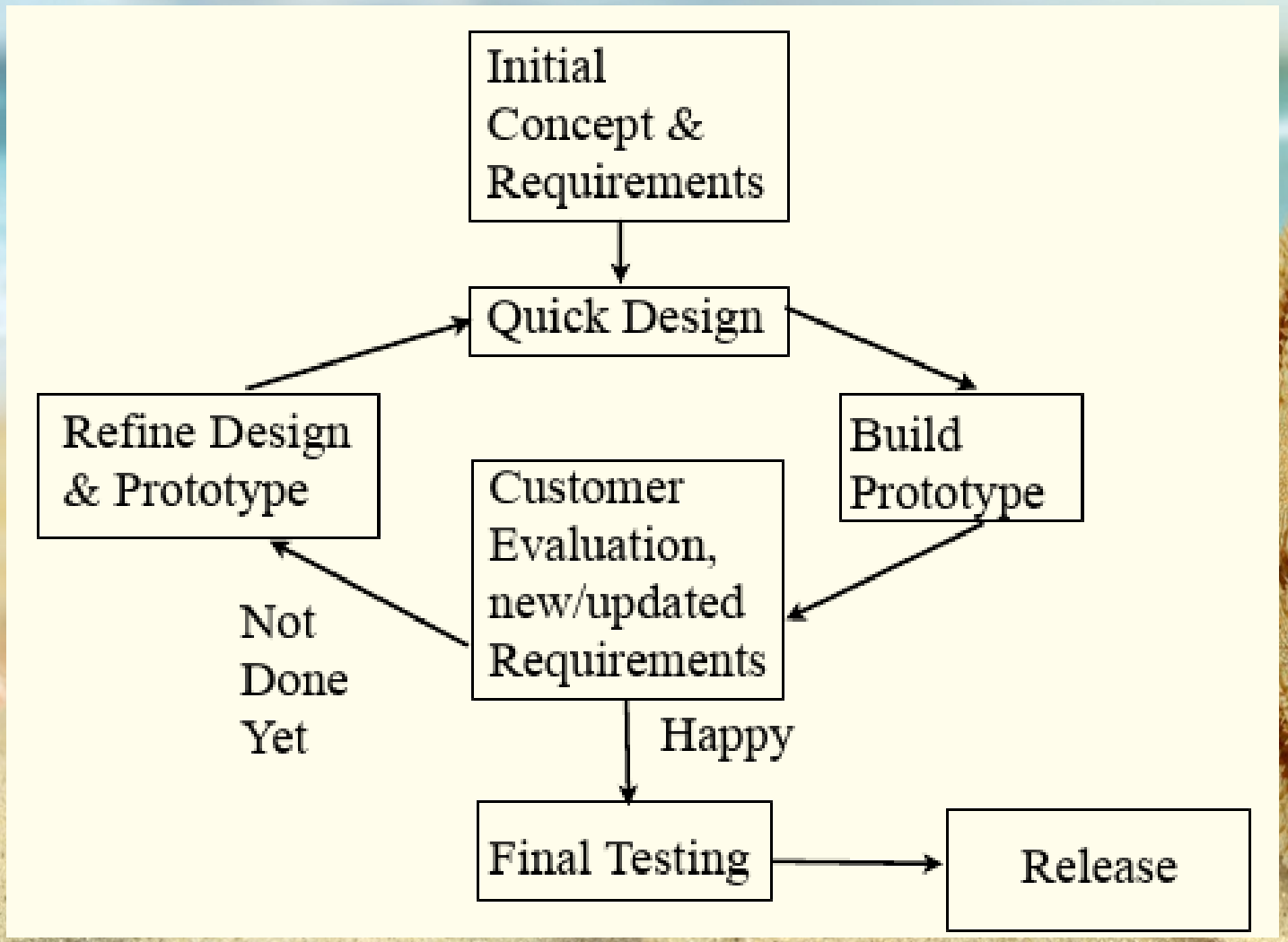
Proyek anda, keseluruhan proyek, mempunyai program yang bisa ditampilkan. Pada jadwal penyelesaian yang dijanjikan, proyek menghasilkan sistem yang diterima oleh user, atau tmungkinjuga tidak bisa diterima user. Setiap orang tahu hasilnya pada hari itu.

Obyek yang membangun proyek membagi proyek dalam komponen-komponen yang masing-masing punya karakteristik yang sama: setiap aktivitas didefinisikan dengan sebuah kriteria penyelesaian dengan hasil yang bisa didemonstrasikan, baik selesai ataupun belum.

Evolutionary prototyping

- Atau disebut Incremental Prototyping
- Memprioritaskan requirements sebagaimana yang didapatkan
- Diperbaiki menggunakan feedback dari user dan testing sampai klien menerima produk
- Cocok untuk requirements yang berubah, atau area aplikasi yang kurang dipahami

Evolutionary Prototyping



Evolutionary prototyping

- Menyadari bahwa memang sulit untuk merencanakan proyek secara penuh dari awal, dan bahwa feedback adalah elemen penting untuk analisa dan desain yang bagus
- Berisiko, tapi mempunyai track record yang bagus

Evolutionary Prototyping

- **Keunggulan:**
  - Progres bisa lebih dilihat
  - Masukan dari klien dan pengguna akhir dalam requirements
- **Kelemahan:**
  - Resiko jadwal, budget, dan ekspektasi progress yang tidak realistis
  - Kemungkinan terjadinya desain yang buruk
  - Kemungkinan kurang maintenabel
  - Mungkin perlu banyak pengerjaan ulang

Leightweight (atau Agile) Models

- Leightweight – sedikit dokumentasi, lebih sedikit kontrol proses
- Untuk proyek kecil hingga menengah (10K – 100K LOC)
- Tim kecil (< 20 developer)
- Semua model agile adalah iterativ

Extreme Programming

- **Mulai dikembangkan 1995**
- **Bagus untuktim kecil (sekitar 10)**
- **Mengandalkan:**
  - Keterlibatan klien
  - Unit testing secara berkelanjutan dan review
  - Pair programming
  - Siklus rilis yang pendek, dan rilis secara berkala

Karakeristik dari eXtreme Programming

- **Planning**
  - Cerita dari pengguna ditulis
  - Rencana rilis untuk penjadwalan
  - Membuat rilis-rilis kecil
  - Proyek dibagi dalam iterasi
  - Rencana iterasi memulai setiap iterasi
  - Komunikasi pada semua anggota grup
  - Meeting singkat mengawali setiap hari

Karakeristik dari eXtreme Programming

- **Desain:**
  - Sesederhana mungkin – redesain untuk memepertahankan tetap sederhana
    - Jalankan semua unit test
    - Tidak ada kode yang terduplikasi
    - Ekspresikan apa maksud kode
    - Sesedikit mungkin jumlah kelas dan metode untuk mengimlpelmentasikan story
  - Pilih metafor sistem: gambaran singkat bagaimana sistem bekerja (bukan arsitektur)
  - Menggunakan CRC card
  - Utamakan refactoring

Karakeristik dari eXtreme Programming

- **Koding:**
  - Klien selalu ada
  - Kode harus ditulis dengan standard yang disetujui
  - Buat kode tes unit lebih dahulu
  - Seluruh kode produksi harus diprogram secara berpasangan
  - Sering-sering melakukan integrasi
  - Menggunakan kode kepemilikan bersama
  - Tidak ada lembur

Karakeristik dari eXtreme Programming

- **Testing**
  - Semua kode harus mempunyai tes unit
  - Semua kode harus lulus uji tes unit sebelum dirilis
  - Jika bug ditemukan, tes segera dibuat
  - Tes penerimaan ditulis oleh perwakilan klien yang di tempat dan dijalankan secara sering dan skornya dipublikasikan

Scrum

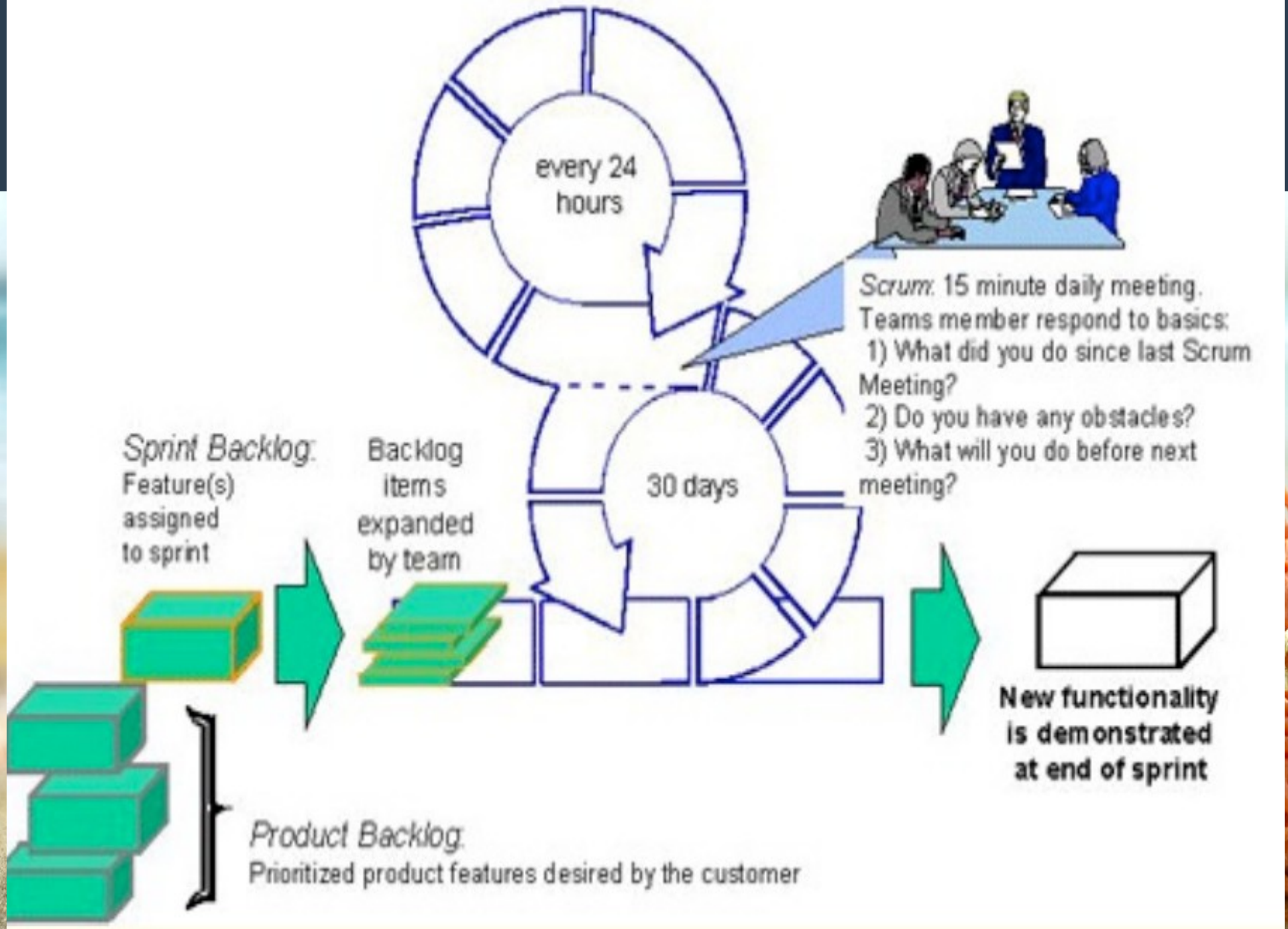
- **Scrum termasuk dalam proses agile**
- **Lebih dulu dikembangkan daripada XP (Extreme Programming)**
- **Menggunakan iterasi singkat untuk mengembangkan produk inkremental**
- **Inti praktek pelaksanaan:**
  - Tim kecil (tidak lebih dari 10 developer per tim)
  - Sering dilakukan build kode
  - Desain dengan sedikit ketergantungan
  - Testing dilakukan secara konstan
  - Kontrol iterasi (time-boxing)

Bagaimana Scrum dilakukan

- **Scrum dicirikan dengan sprint**
  - 1 – 4 minggu iterasi
  - ScrumMaster melakukan ScrumMeeting setiap hari
    - Tiga pertanyaan:
      - Apa yang sudah anda selesaikan sejak pertemuan yang lalu?
      - Apa yang menghambat dalam menyelesaikan pekerjaan
      - Apakah yang akan kamu kerjakan sampai pertemuan berikutnya?
- **Sprint harus menghasilkan produk yang berjalan**
- **Jadwal deadline tidak ditunda, jika ada fungsi software yang belum selesai, maka program akhir pada deadline dikurangi fungsionalitasnya**
- **Requirement dirangkum dalam dua backlog:**
  - Product backlog berisi daftar requirements prioritas untuk proyek, dibuat tim scrum dan pemilik produk
  - Sprint backlog berisi daftar prioritas yang harus diselesaikan oleh sprint yang berjalan
-

Aturan Sprint

- **Fokus total**
- **Tidak boleh ada interupsi/perubahan dari luar**
- **Pekerjaan baru mungkin ditemukan oleh tim (dan hanya boleh oleh tim tersebut)**
  - Pekerjaan ini dimasukkan ke dalam sprint backlog


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Tahapan Sprint pada Scrum

- **Sebelum sprint pertama dimulai, dilakukan fase inisialisasi untuk membuat requirement awal, penentuan arsitektur untuk mengimplementasikan requirement, membagi story dari user ke dalam grup sprint, dan membagi sejumlah story dari user ke dalam pekerjaan yang harus diestimasikan pengerjaannya.**
- **Penyiapan sprint ini tidak boleh melebihi 1 hari**
- **Penyiapan sprint berakhir jika semua estimasi pengerjaan sudah memenuhi waktu yang teralokasi untuk sprint**
- **Setelah setiap sprint, dilakukan lagi pertemuan untuk merencanakan sprint berikutnya.**
  - ScrumMaster dan tim melakukan prioritasi ulang dari product backlog dan membuat backlog untuk sprint berikutnya

Setelah akhir sprint yang dijalankan, dilakukan final sprint untuk menutup proyek, untuk menghasil rilis produk akhir

Keseluruhan yang dilakuka pada agile process

- Mendaftar requirement awal (menggunakan kertas untuk prototip, atau use case/ scenario)
- Memprioritaskan requirement
- Menentukan siklus iterasi (3 – 5 minggu; lebih baik 1 – minggu)
- Menentukan requirement untuk siklus berikutnya
- Gunakan requirement tersebut untuk mendesain produk pada level atas
- Tentukan requirement dari keseluruhan requirement untuk mendesain komponen
- Bagi proses desain komponen dalam sejumlah kelompok pengerjaan
- Bagi lagi kelompok pengerjaan itu dalam job-job kecil
- Estimasikan pengerjaan setiap job kecil tersebut, jika melebihi 8 jam (satu hari kerja), pecah lagi dalam pekerjaan lebih kecil dengan masing-masing waktu pengerjaaan kurang dari 8 jam
-

Agile Process

- Kelompokkan daftar pekerjaan-pekerjaan dengan prioritas tinggi yang bisa dilakukan dalam satu siklus iterasi.
- Tulis dan uji kode untuk semua pekerjaan pada siklus yang sedang berjalan
- Rilis hasil dari siklus yang berjalan pada klien, yang nanti harus melakukan uji penerimaan
- Perbaiki bug pada uji penerimaan
- Selesai acceptance test (uji penerimaan) mulai lagi pada step untuk memulai siklus selanjutnya
- Jika sudah selesai dengan pekerjaan yang direncanakan atau sudah mencapai deadline, rilis produk


Semoga
Bermanfaat