Managing Architectural Change

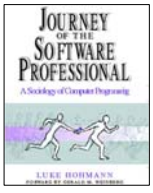
… or, how to get product management to support rewriting the system in Ruby

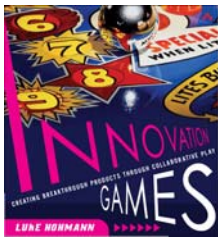
(am I kidding or not)

Motivated from Within®

About Luke Hohmann

- Consultant, manager, **coach**
- Unique perspective, motivated and informed from variety of
  - solutions
  - roles
  - companies





What is "Software Architecture"?

"A system architecture defines the basic 'structure' of the system (e.g., the high-level modules comprising the major functions of the system, the management and distribution of data, the kind and style of its user interface, what platform(s) will it run on, and so forth)."

[Hohmann: *Journey of the Software Professional*, 1996]

Why Does "Software Architecture" Matter?

- Longevity
  - System longevity: 12 – 30+ years
  - Developer longevity: 2 – 4 years
- Degree/nature of change
- Profitability
- Social structure
- Boundaries and dependencies

Alternative Viewpoints

- Organic vs. inorganic models
- Functions vs. Capabilities
  - Function: What a product should do
  - Capability: The underlying architecture's ability to support a related set of functions
- Dependency management
- People management / “wholeness”

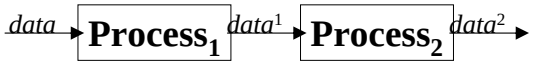
Architecture Styles/Patterns

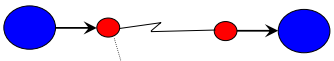
Transaction Management

Domain Model

Object Translation

Persistent Store



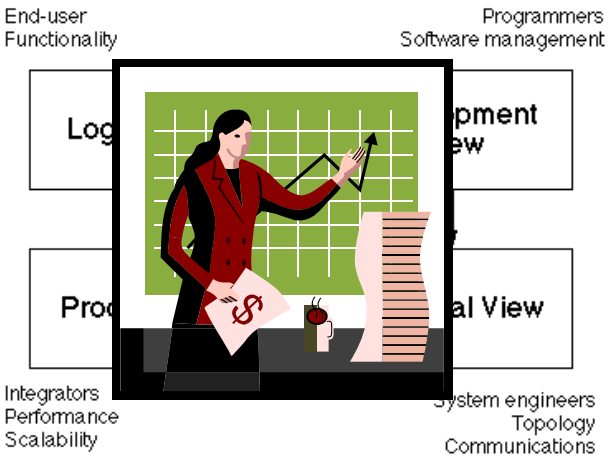


Distributed Proxy

Try This Now...

- “Draw the architecture” test
- Can you enumerate the capabilities of your current architecture?
- What’s the difference between local / non-local / architectural change?

n+1 View of Architecture



Try This At Work...

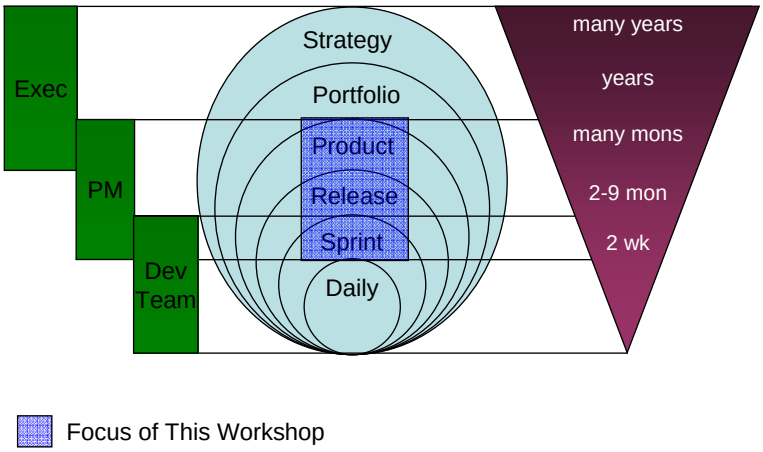
- Do the “Draw the architecture” test with your team. Post the results on a wall. What happens?
- Which of the *n+1* views of architecture does your team think are most relevant to your system?
- Which of these views are you willing to maintain over the duration of the project?

Agile Development Processes

Agile is all about planning for and participating in the infinite game

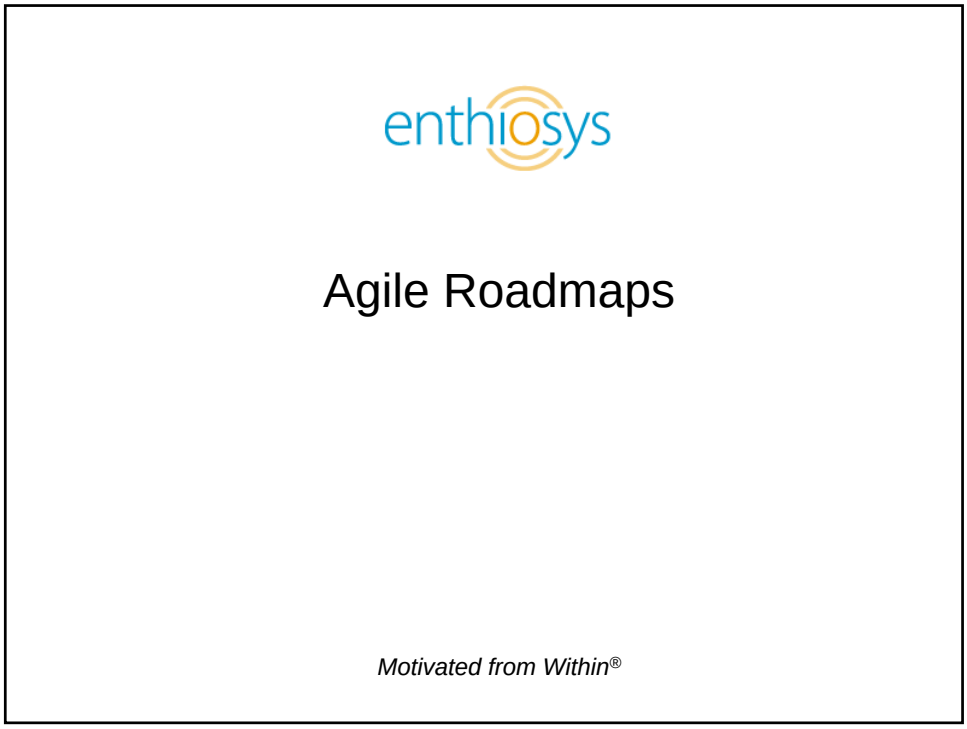
Motivated from Within®

Context: Planning Time Horizons



Agile Planning Artifacts

1.	Vision / #1@	2-4 years
2.	Roadmap	who, when, what, why
3.	Release Plan	# & theme of iterations
4.	Iteration Plan	tasks for the iteration
5.	Daily Scrum	yesterday, today, blocks



Agile Roadmap

enthiosys

- A living document designed to answer key strategic questions:
  - Who are my desirable markets/market segments? What do they care about?
  - When / how often should I serve them?
  - What technologies can I leverage? How must my current product change to deal with the answers to these questions?
  - What are the external factors that I must address to deal with these issues?

Benefits of a Roadmap

- Roadmaps identify and clarify the tactical and strategic intent of your product
- Internally:
  - Becomes the filter for backlog prioritization
  - Gains consensus around direction
  - Ensures the “ship is headed in the right direction”
  - Avoids the “last/loudest” priority problem
- Externally:
  - Provides customers with access to near-term commitments and long term “points of view”
  - Binds customers to your company

Typical Roadmap Failures

- No visible logic
- Created unilaterally
  - Lack of buy-in
  - Poor technical and market inputs
- No plan for internal or external sharing

Successful Roadmaps Creation…

- Active participation of key constituents
  - Engineering (architects), Marketing, Support, next-level product strategists
- Extended in-person meetings
- Time to research issues
- Quarterly reviews
- Clear (written) distribution plan
- *Easy to say, hard to do*

Low-Tech Speeds Collaboration

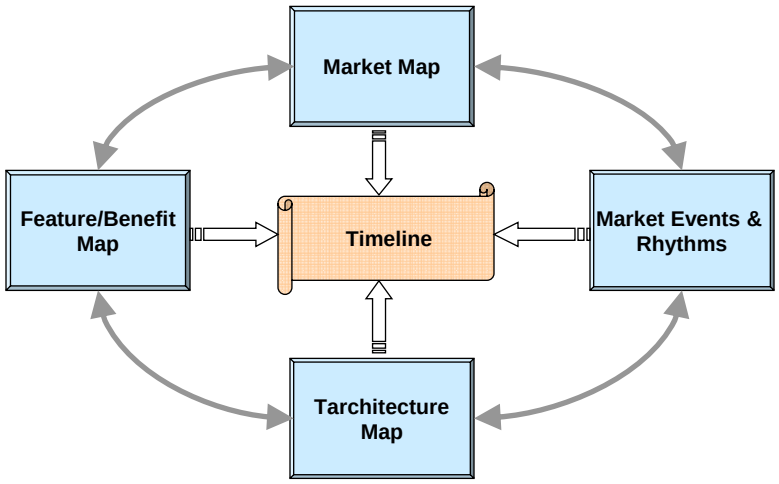
Formal results can be transcribed in various tools

Roadmaps are Scary

- Forces you to articulate what you are ***not*** going to build
- Makes you commit to an uncertain future

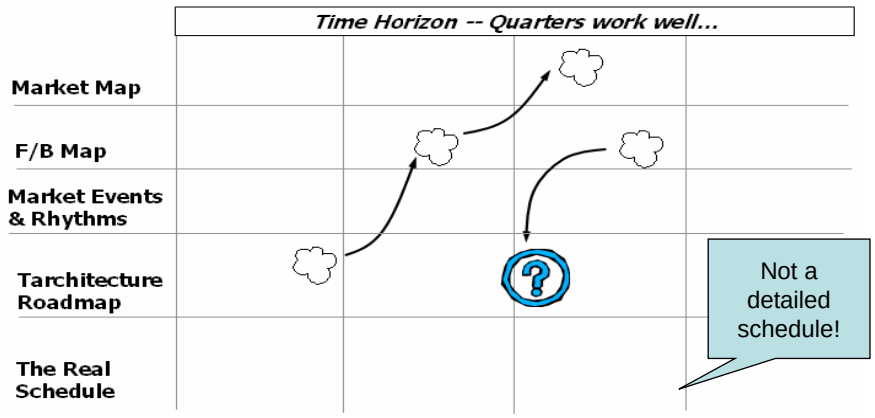
19

Creating a Strategic Roadmap



20

An Approach to Roadmap Layout



Enthiosys is "Template Agnostic"!
Use a format that works best for you…

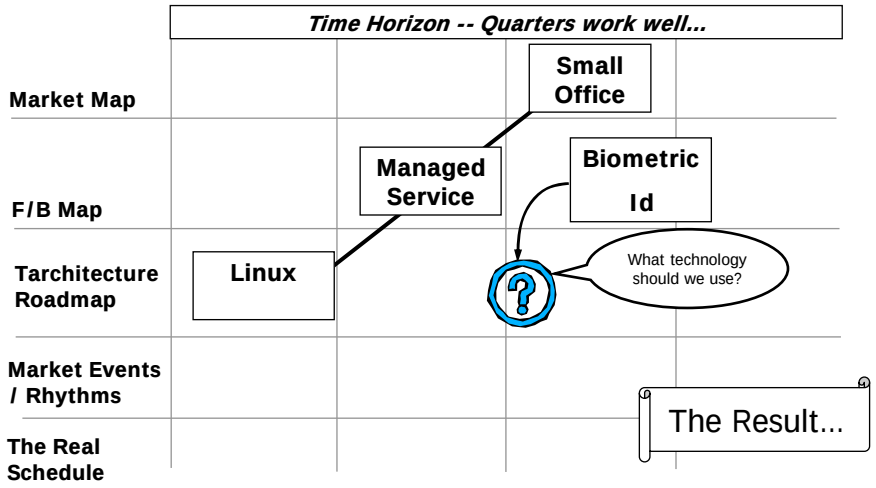
21

Case Study: Acquisition Digestion

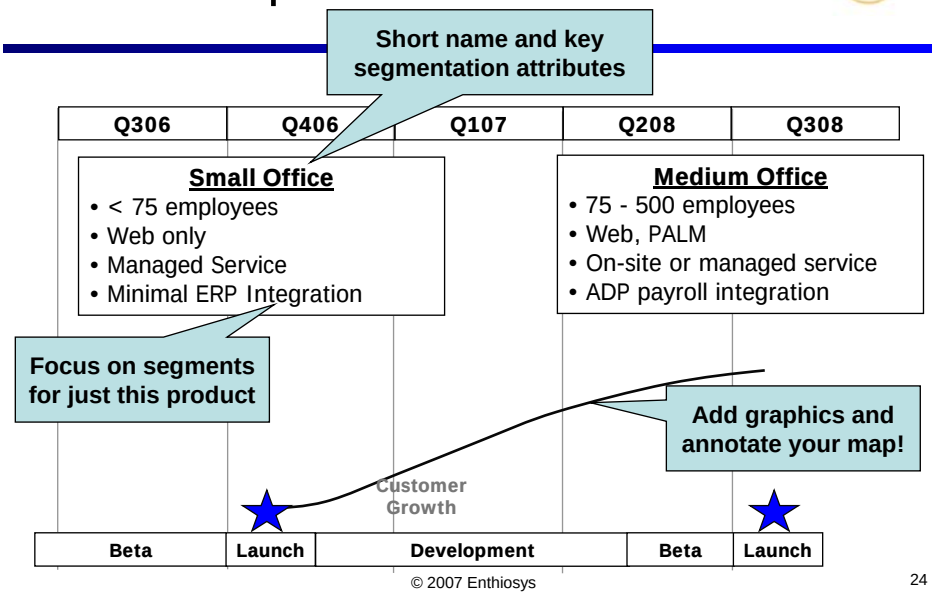
- Startup offering self-service access to small company benefits systems via "smart phones"
  - Want to support multiple devices
  - Innovator-adopters like the system but are frustrated with the slow frequency of releases
  - Your unsure if system should be deployed as a service or as a customer-premise
  - Customers have asked for backend integration
  - Development wants to leverage more devices

22

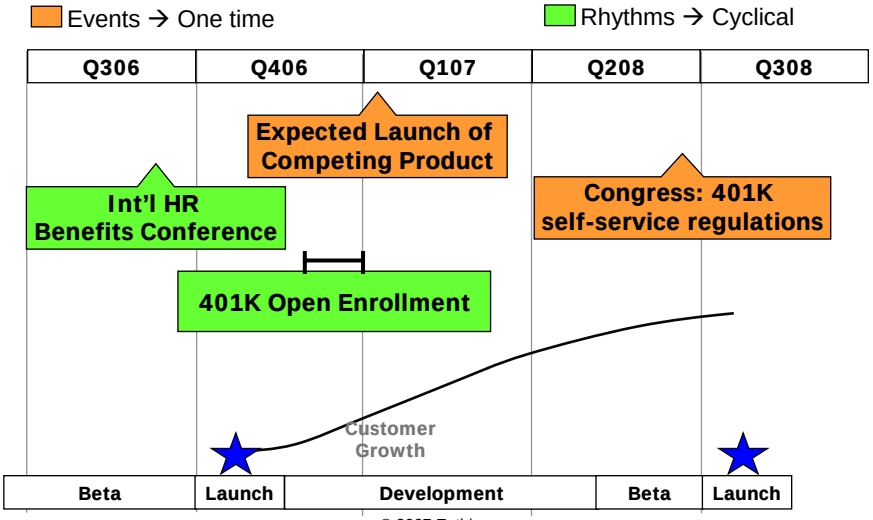
Build Your Strategic Roadmap Iteratively



Market Map

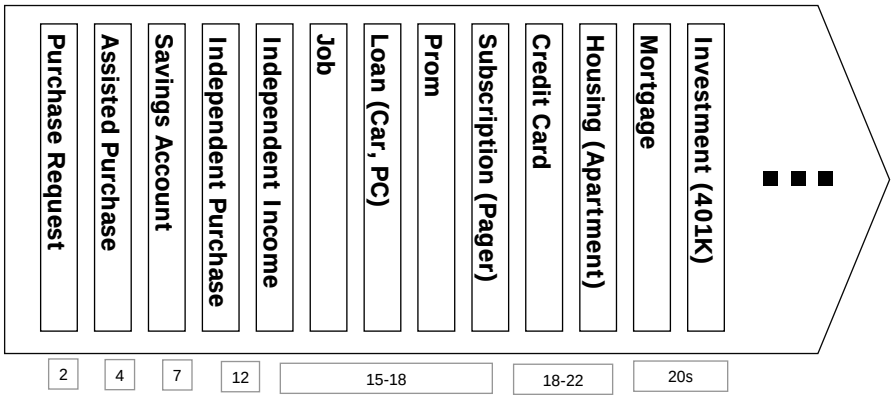


Market Events / Rhythms



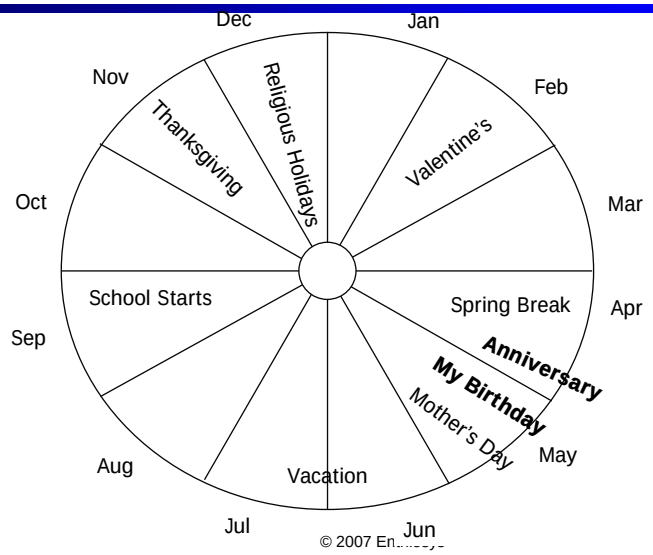
25

Life "Firsts"



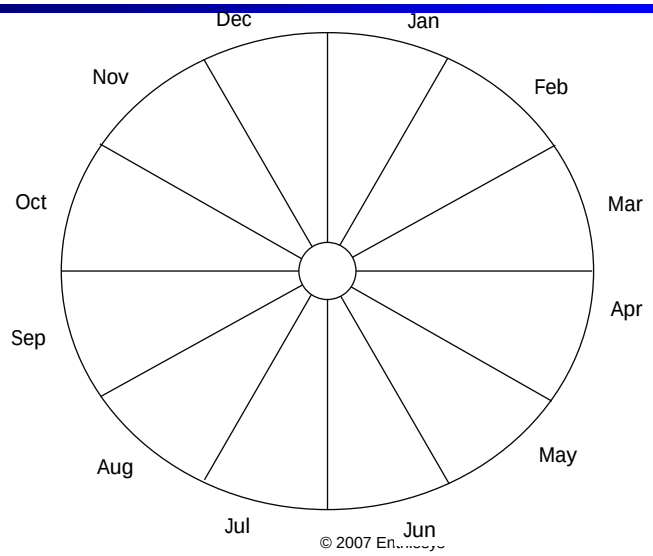
26

American Life Rhythms



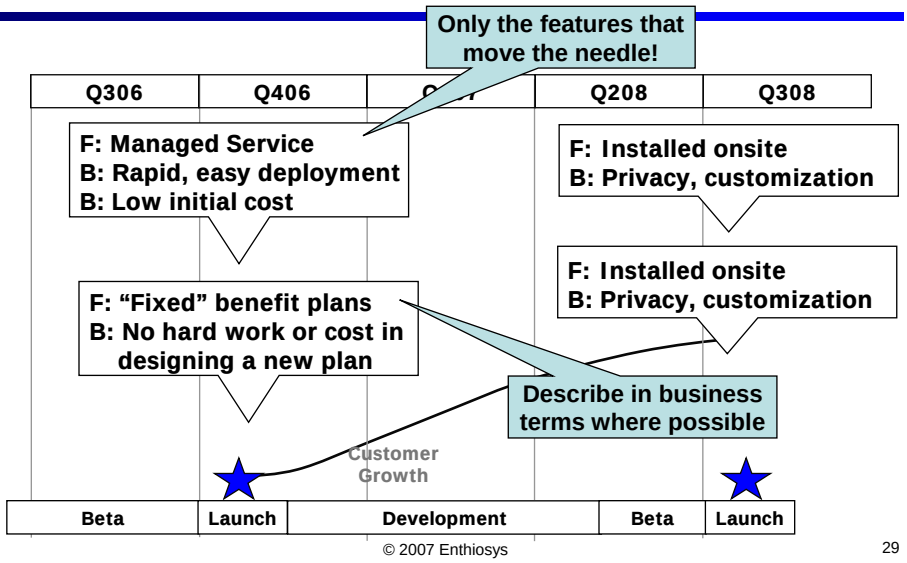
27

Market Rhythms



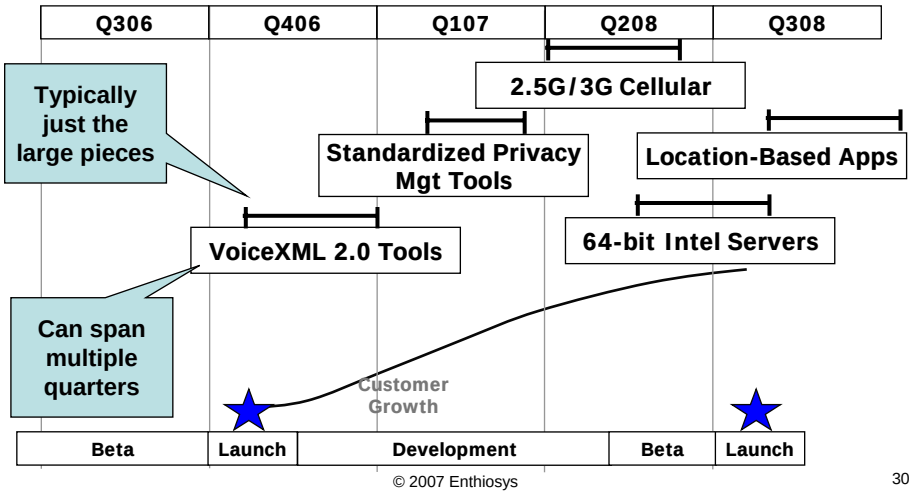
28

Feature / Benefit Map



29

Tarchitecture Roadmap



30

Focus on Capabilities

- Major components of the application that enable multiple features
- EX: Notification Engine
  - Allows users to send schedule, event, or conditional based notifications via any communication channel.

Managing Architectural Change

- SEI classifies changes as:
  - Local — fix a bug in a module
  - Non-Local — add new features within existing architecture
  - Architectural — swap out a user interface library with a new library
- To manage architectural change
  1. Lodge the change into the roadmap
  2. Ensure it is on the backlog
  3. Ensure it is prioritized into actual work

Scheduling Considerations

- Holidays & vacations
- Internal events & rhythms
  - Quarterly earnings calls
  - Peer reviews
- Customer commitments
- Other milestones

Discussion

- What occurred for you as you were creating:
  - Your market map
  - Capturing market events/rhythms
  - Capturing features and benefits
  - Identifying technical opportunities
- What might be your next steps?

enthiosys

Managing Your Backlog

Motivated from Within®

Managing The Backlog

enthiosys

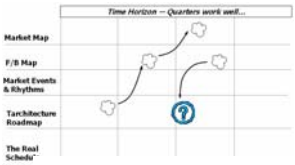
- Your job is to define and prioritize customer value through the backlog

markets to serve

money to make

pile of stuff to do

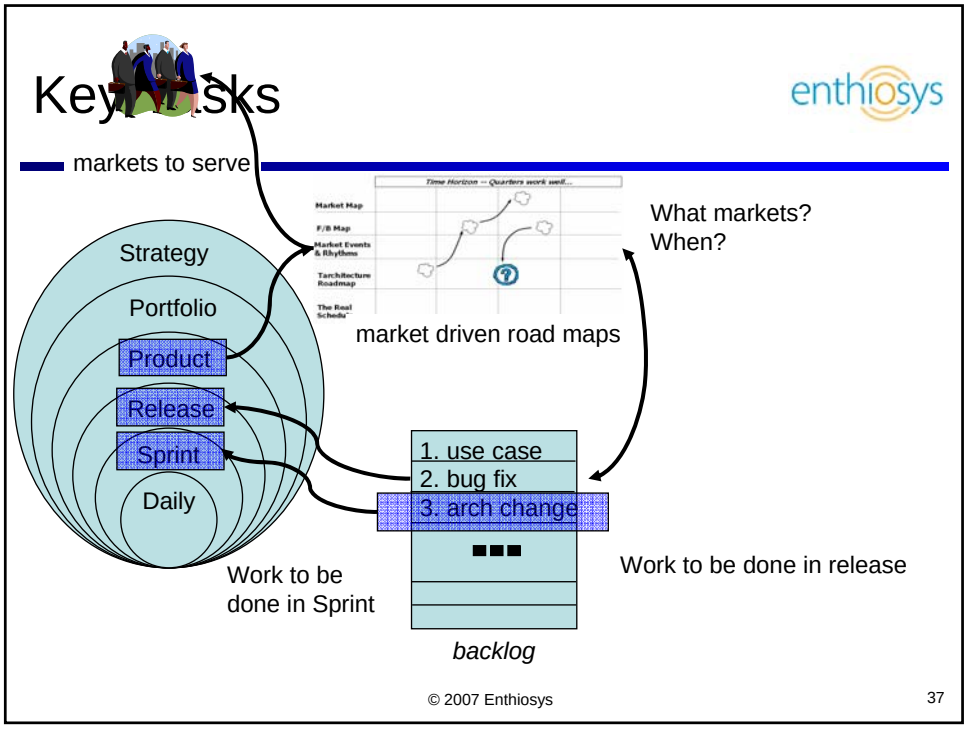
- use cases
- bugs
- features
- enhancement requests
- updates
- stuff… to do…

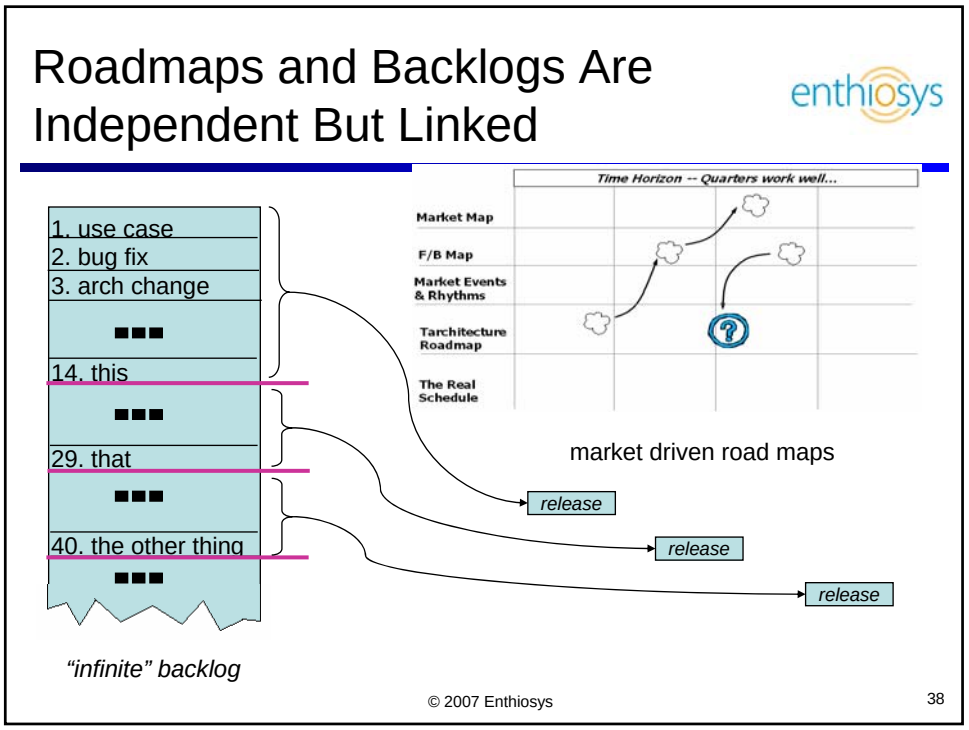


market driven road maps

1. use case
2. bug fix
3. arch change
■■■

backlog


Key Tasks
enthiosys
markets to serve
What markets?
When?
Strategy
Portfolio
Product
Release
Sprint
Daily
market driven road maps
1. use case
2. bug fix
3. arch change
Work to be
done in Sprint
Work to be done in release
backlog
© 2007 Enthiosys
37


Roadmaps and Backlogs Are Independent But Linked
enthiosys
1. use case
2. bug fix
3. arch change
14. this
29. that
40. the other thing
Time Horizon -- Quarters work well...
Market Map
F/B Map
Market Events & Rhythms
Tarchitecture Roadmap
The Real Schedule
market driven road maps
release
release
release
"infinite" backlog
© 2007 Enthiosys
38

Legacy Code is Crappy

- Many times code is never what you want
- Ensure that new code *is* what you want
- Fix old code incrementally
- BEWARE of Lunde's law:

> "Any time the senior developers on a project are replaced by a new team, the new team will eventually find a compelling reason for a total rewrite."