beyond
nerds bearing gifts

the future of the open source economy

Thank you all for having me here,
this is my first time in Australia,
and it has been an amazing experience so far.

the future of the open source economy

beyond nerds bearing gifts

I was originally expecting more of a culture shock,
coming from the northern hemisphere,
and I even prepared <X> some custom Australian slides,
but it turns out I was misled,
the equipment is actually compatible.

Anyhow, I have a problem
which I'd like to share with you,
and it's not an uncommon problem.
It's probably one many of you share.

Erica

My problem is, <X>
I have a mother in law.

And when we sit down to Christmas dinner,
which is coming up,
she has this question she asks,
and it is a perfectly reasonable question
for a mother-in-law to ask a son-in-law.

“what is it, that you *do*?...”

She asks me, <X>
what it is that I "do".
What do I "do"?

The subtext being,
am I doing something that can rationally be expected <X>
to help support her perfect daughter
and two darling grandchildren?

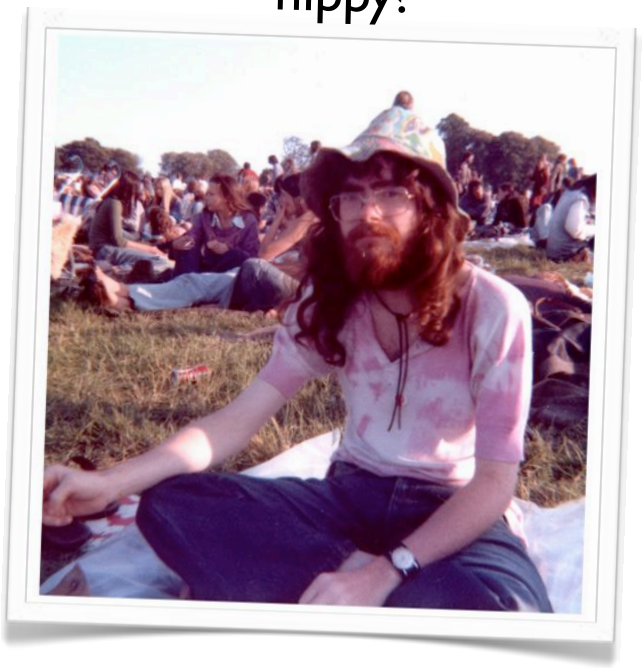
And this is where things get difficult.
Because what I "do" is,
<X> I sit in my office
at the back of the house,
and I work on software
that people are encouraged
to download and deploy and use
for free. For free.

She understands the programming part of it well enough,
at least insofar as I sit in front of my computer and I type things on it,
but the "for free" part causes serious discomfort.
Because how do the perfect daughter
and the two darling grandchildren get supported "for free".

Why am I confident
that I can earn a living in my chosen field,
when my work product is free?

crazy?

hippy?



Am I <X> crazy, perhaps?
Or some kind <X> of utopian hippy?
Maybe not good son-in-law material.

The proximate answer --
and on a good night this will cool her off --

is that I'm employed by
a company, <X> OpenGeo, that
pays me to work on this
software that we give away for free.
Voila!
But it doesn't take much to see past that dodge,

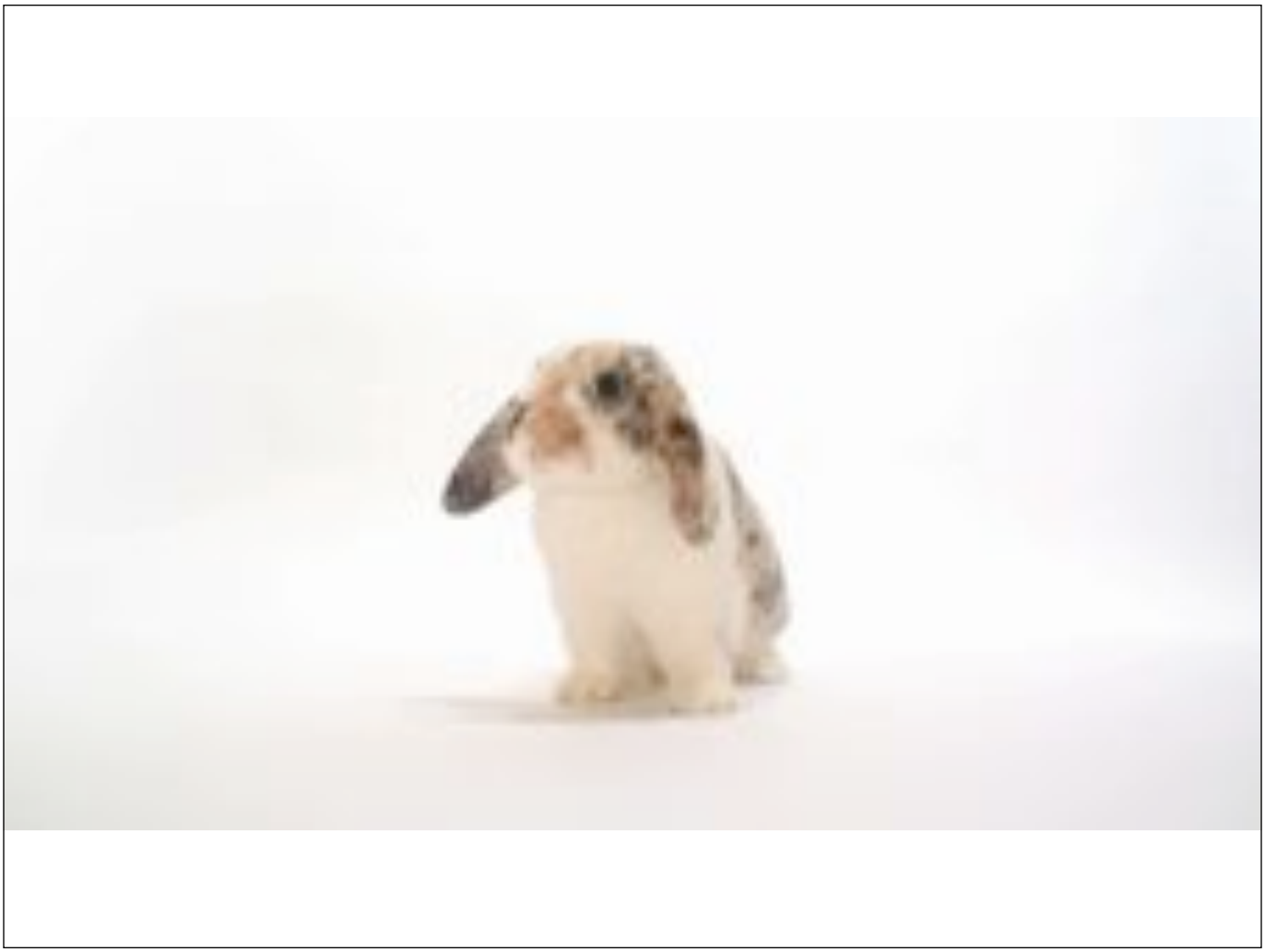
"... then, who pays OpenGeo?"

"Doh!"

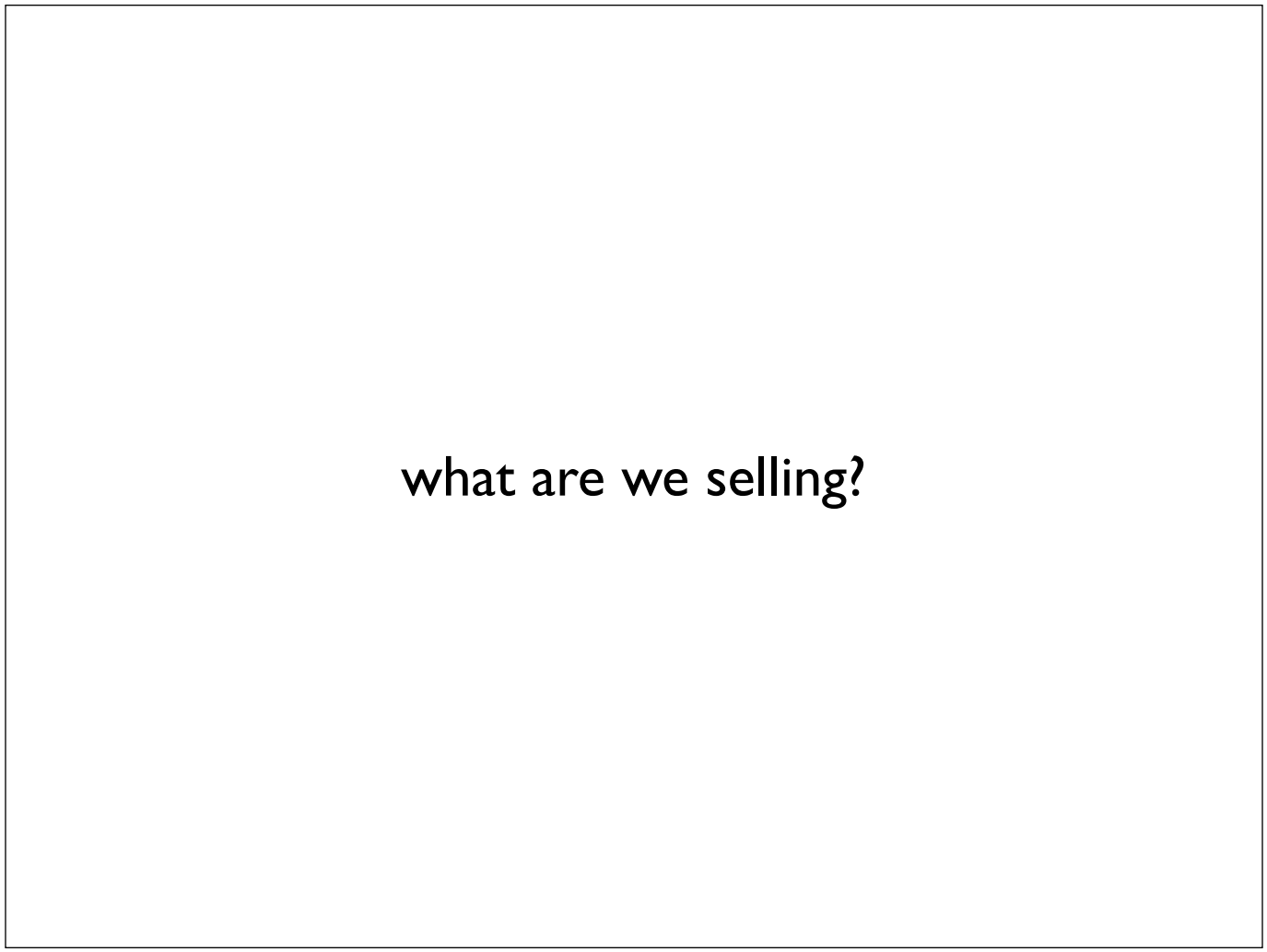
and by her second glass of wine she'll come around again,
"so, <X> who pays OpenGeo for this free software?"

Who indeed?

In the competitive marketplace,
<X> full of beasts like Oracle, Apple and Microsoft
(that picture of Steve Balmer freaks me out)
how,



<X> how can something
as warm
and fluffy
as an open source company
survive?



What is it,
exactly, <X> that we are selling?

If you want to understand
what open source companies are selling,
it helps to understand what the
existing proprietary vendors are selling,
and --
here's the surprising part --

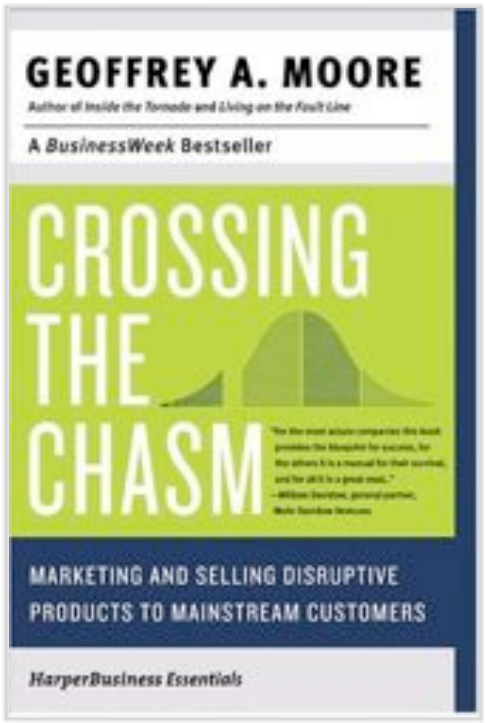
proprietary companies
aren't selling **software**

they are selling **products**

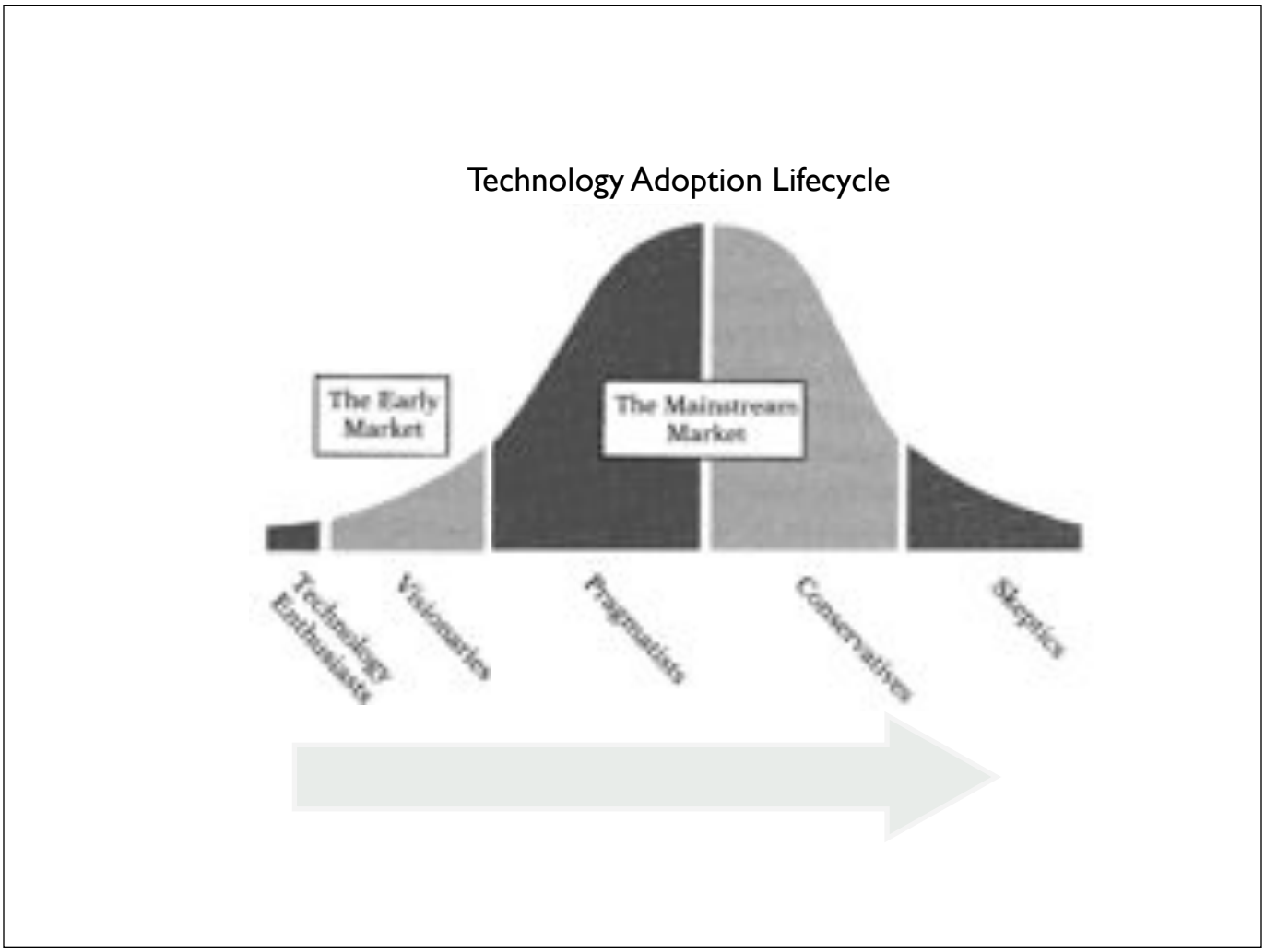
<X> they aren't selling software.

<X> They are selling *products*.

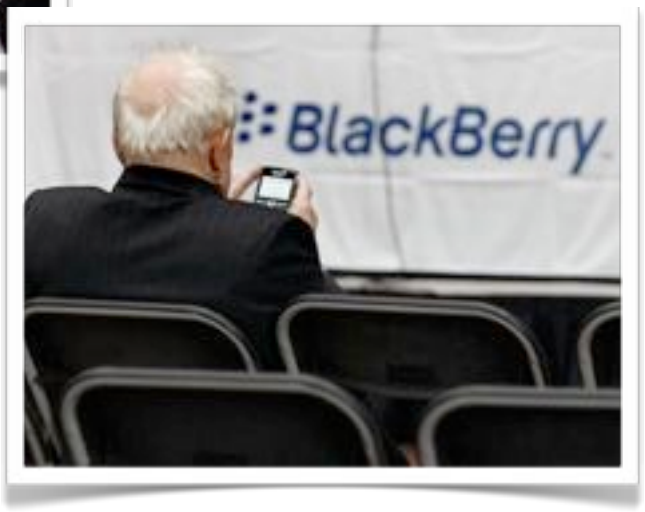
Let me explain what I mean by that.



<X> In the business classic "Crossing the Chasm"
(which I highly recommend),
Geoffrey Moore says that in the
technology adoption life cycle, <X>

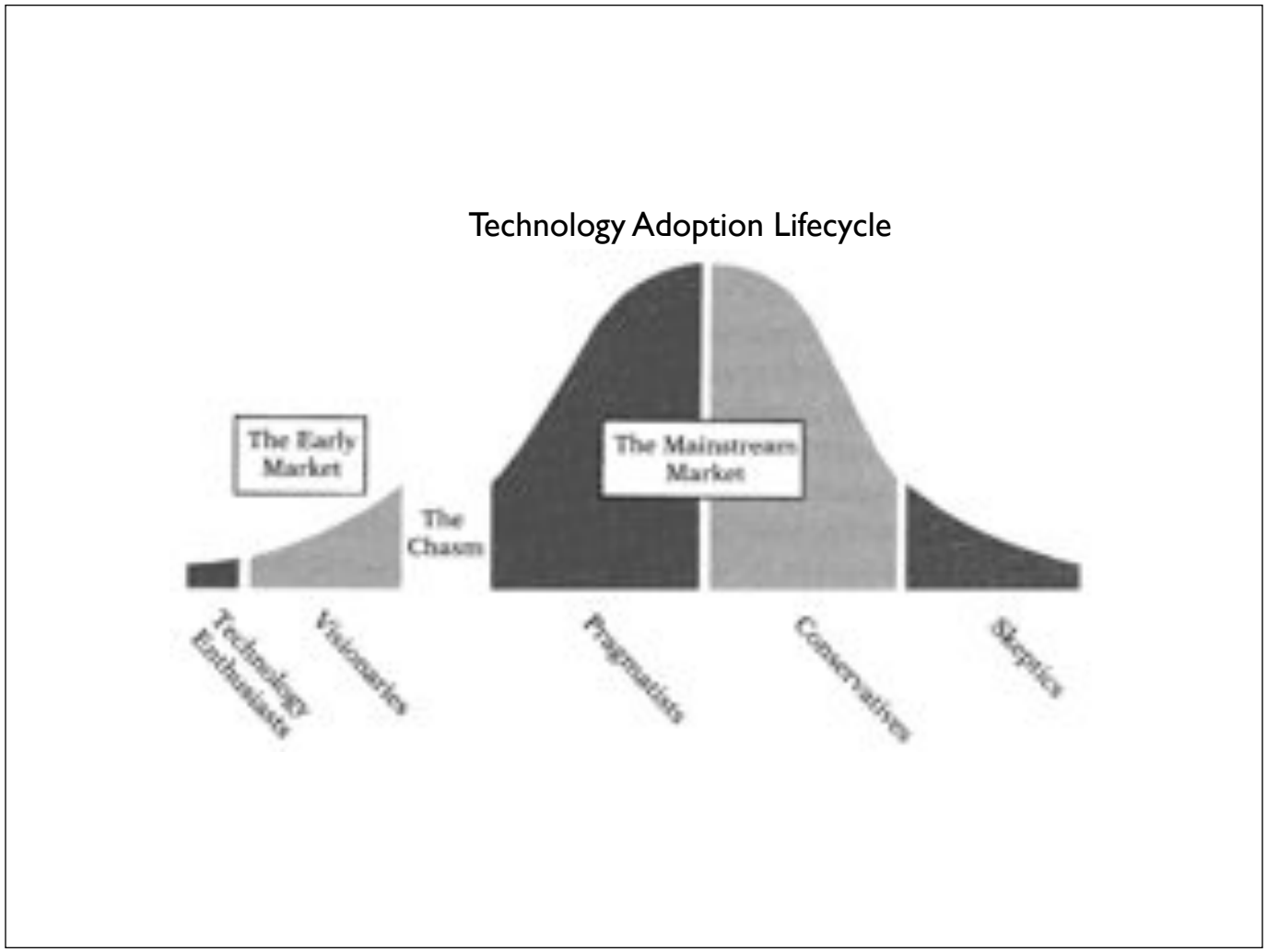


(which is traditionally understood
as a smooth passage
from the small early market of
visionaries and early adopters
to the large mainstream market
of pragmatists and conservatives),
there is
a little understood gap,
in fact a huge *chasm*,
<X> between the small early markets,
and the big mainstream markets
And this chasm is there
because the personalities
of customers in the
early market
are very different
from the personalities
of customer
in the mainstream market.



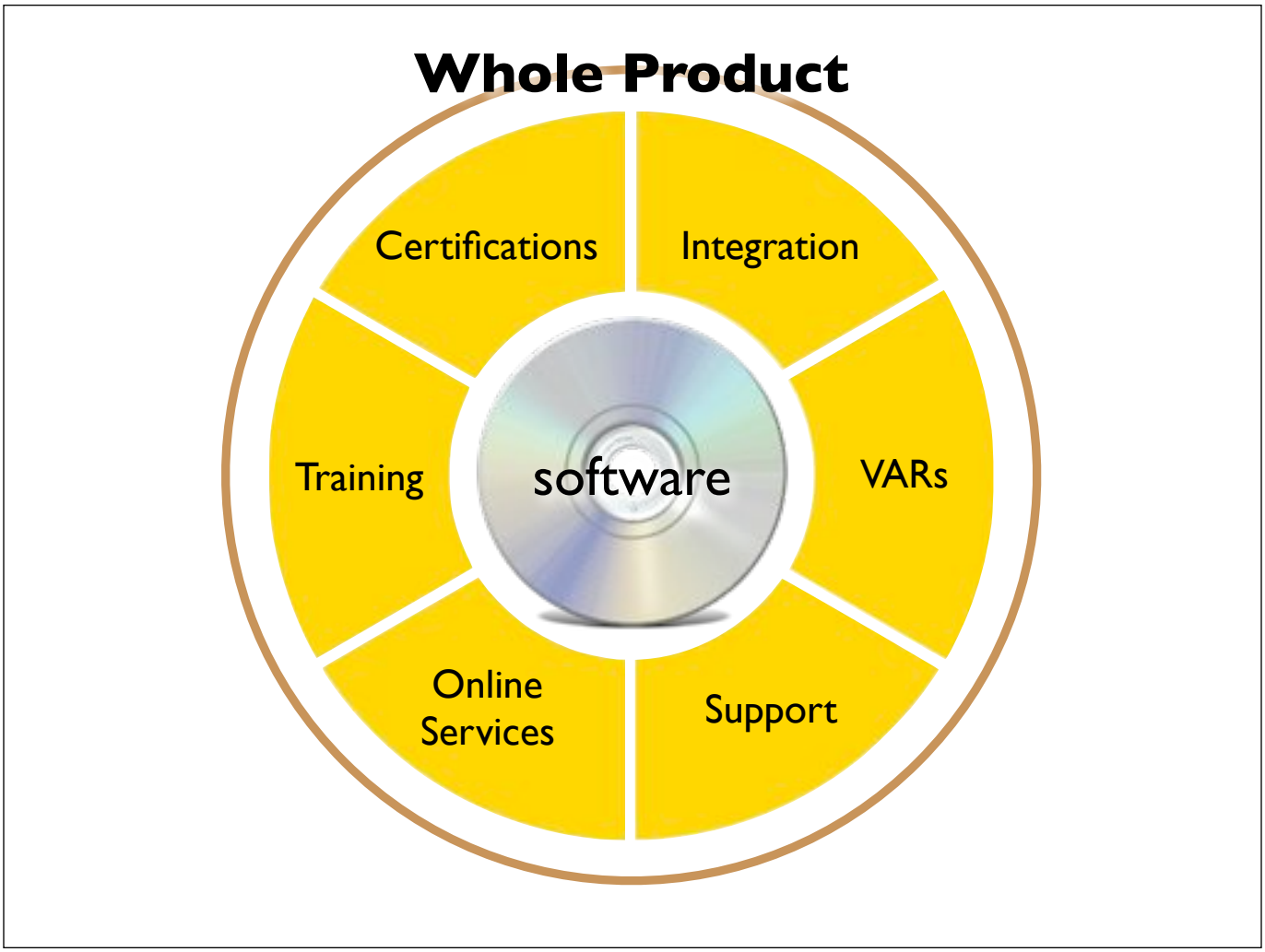
Early adopters and visionaries
have a high tolerance for risk
They like to learn things themselves,
and don't need a lot of support. <X>
Here's an early adopter
with the iPhone he bought on the
first.
day.
they were available.

The <X> mainstream customers are exactly the opposite.



In order to prosper, growing software companies <X>
must cross the market chasm,
to gain access to the
big mainstream markets,
and to do so,
Moore says they must transition
from just selling software

into selling a what he calls <X>
a "whole product".



Now, a young naive technology company might say
"but we have a product! <X> it's on this CDROM right here!"
But they don't have a product, they have <X>
software. What they have is
salable in the early markets,
but *not* to the majority markets.

The "whole product" has software at its core,
but it adds in a critical layer
of extra services and infrastructure around the outside,
Things that reduce the risk (or perceived risk)
associated with adopting the product.
<X> Training courses, support infrastructure,
re-sellers and consulting networks,
update mechanisms, and so on.

And it is the combination of the software
with the added layer of valuable extras
that make a <X> compelling "whole product".

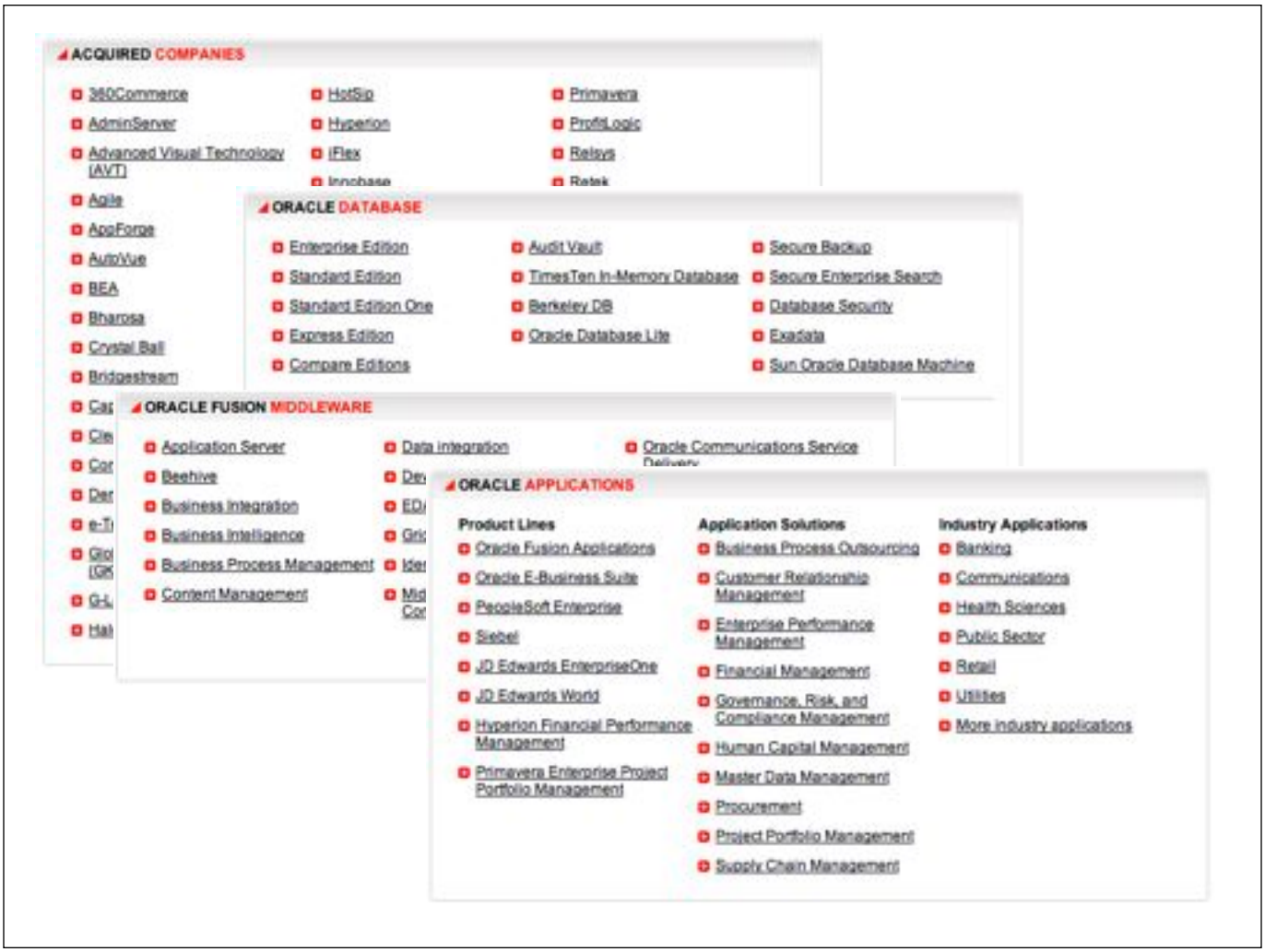
Only a "whole product" can move
from the adventurous early adopter market
into the risk-averse mainstream market.

For example, <X> as a piece of database *software*,
Oracle is not all that compelling.
It's kind of bulky, it's very hard to learn,
and it's pretty easy to screw up.

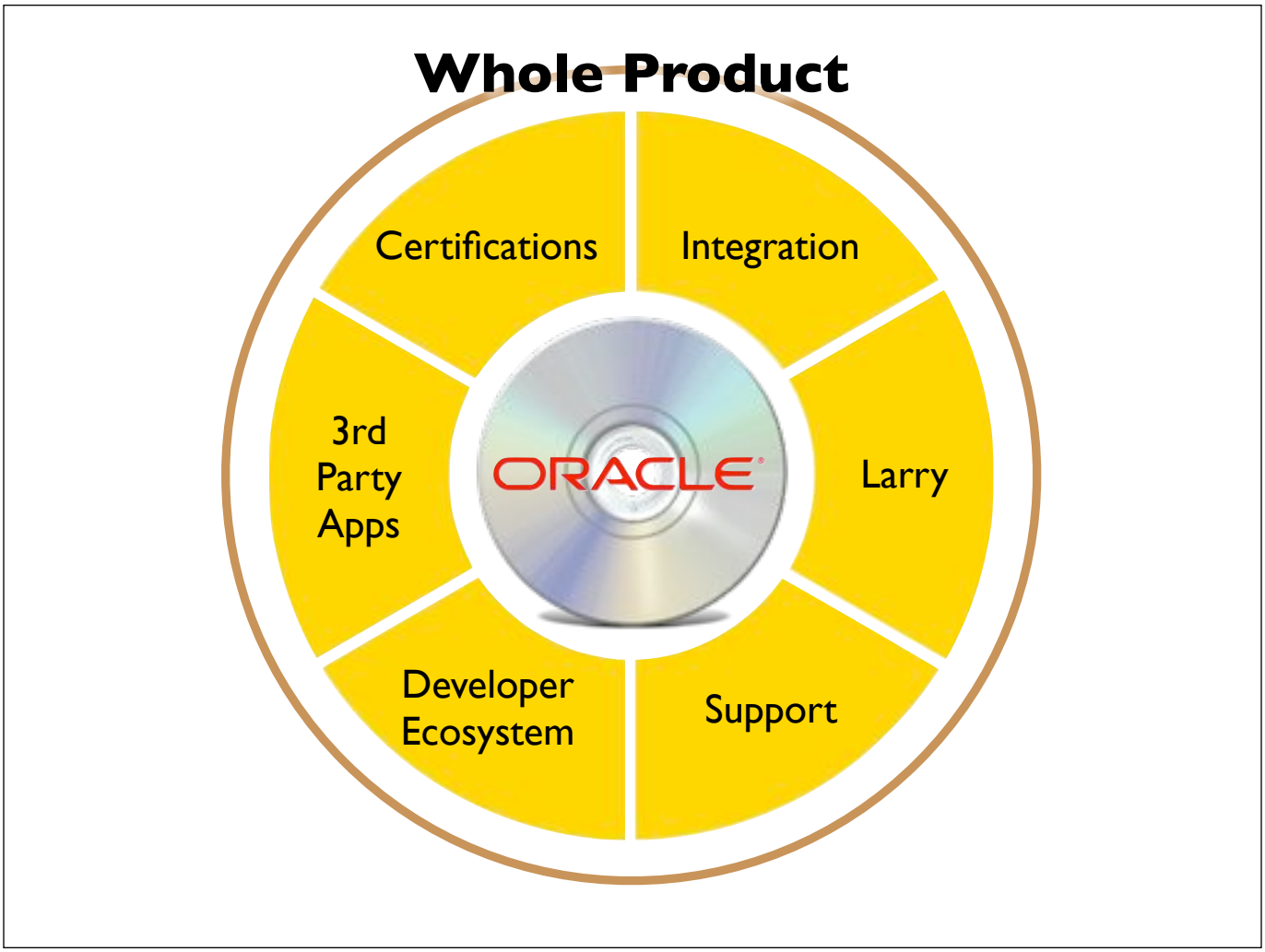
But, add in
<X> a 300lb shelf of professionally written documentation,
training to build a population of developers and administrators,

<X> a rich ecosystem of third party tools,

<X> a reputable (if mercenary) company providing support,
<X> an evil genius,

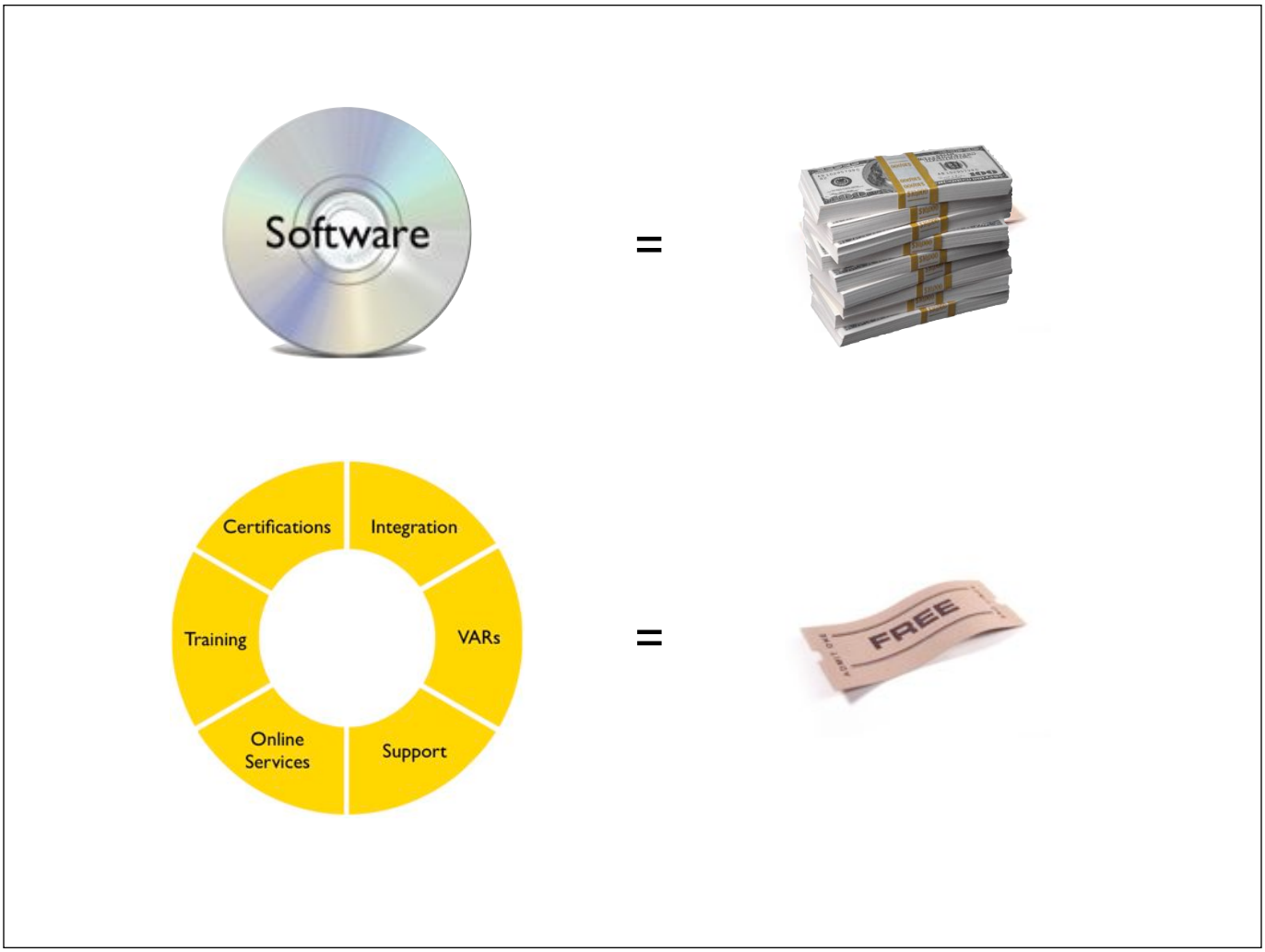


and <X> deep integration with the
other elements of the Oracle software portfolio,



and you have <X> a very compelling *whole product*.

Here's an odd thing.
Even though it is easy to see
that the "whole product" offers
tremendous value beyond the software itself,



our <X> mental model of technology acquisition
is still one where we pay money for the software,
and the vendor throws in
the rest of the whole product
for free.

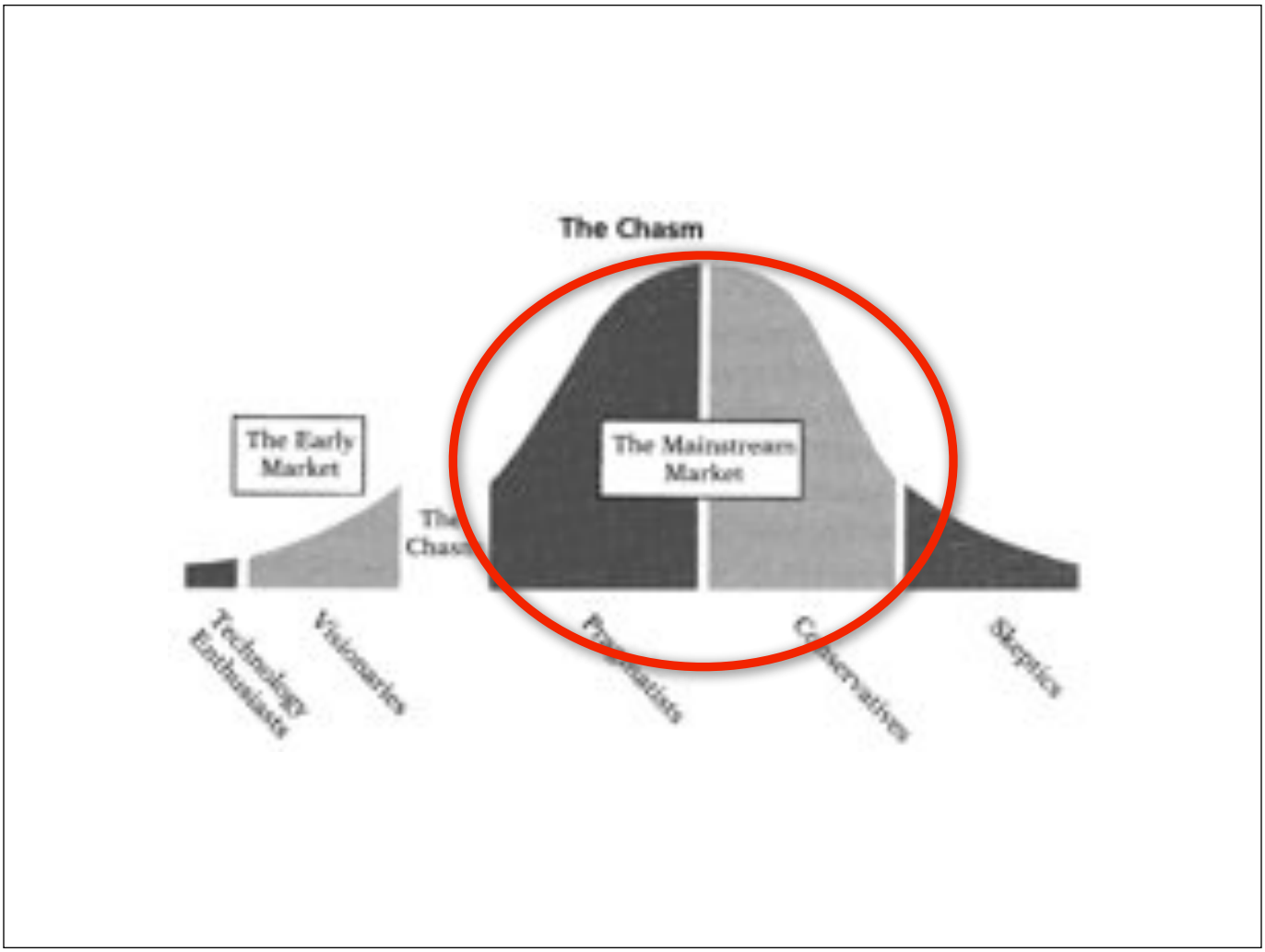
All the great extra value, for free.
And this is where our cognitive dissonance
about open source software companies comes from.
Because what happens
to our mental model
when the cost of the software <X>
goes to zero?

“give me the software...”

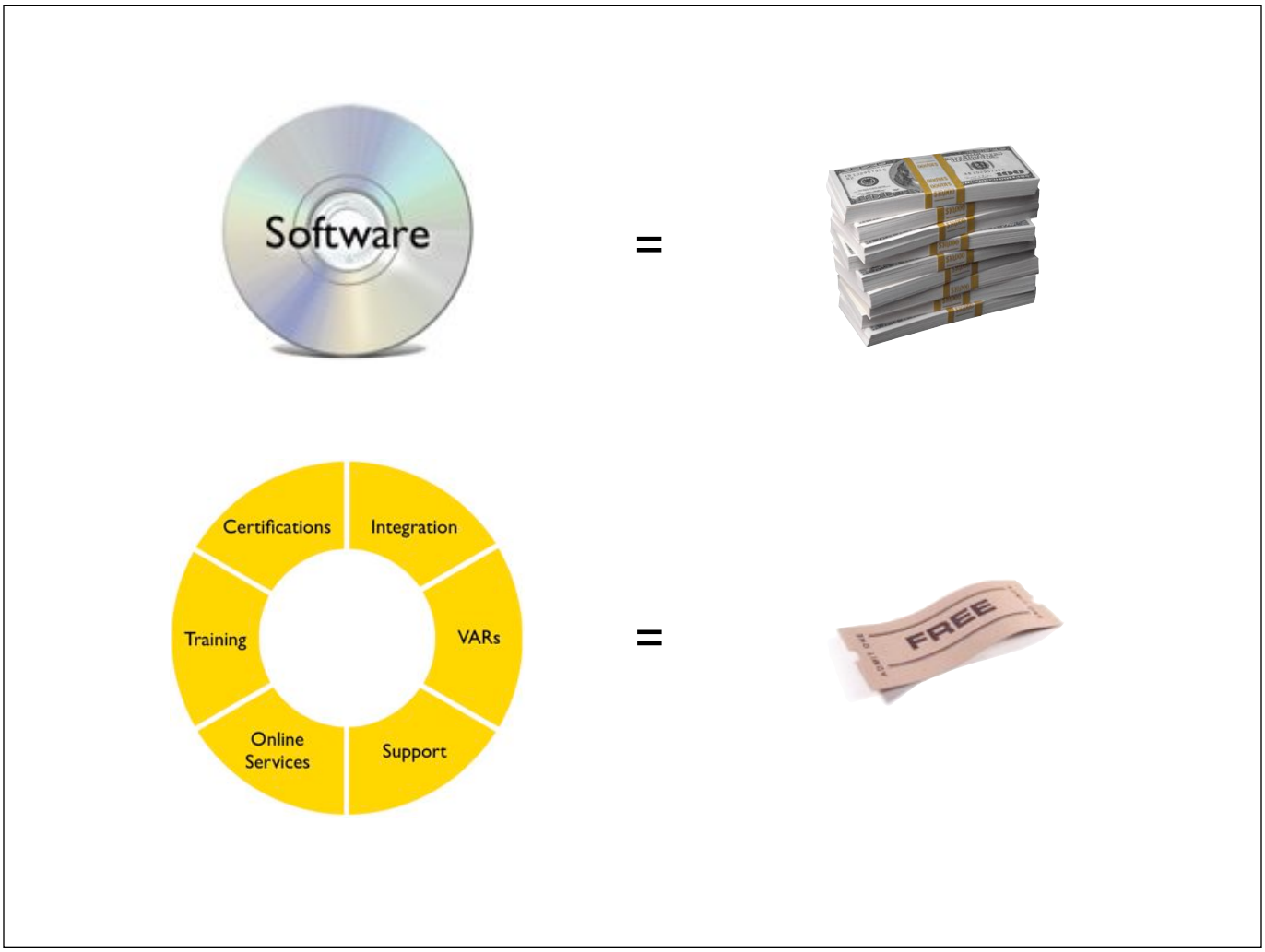
“... and then leave me alone!”

No company can give away the software for free,
and also provide the rest of the
whole product for free! Something has to give!

For the early adopters and technology visionaries,
the people in the early market,(and lets be honest,
this audience is full of you people)
there's no problem -- <X>
they just use the software as is,
and take advantage of the thinner layer of
support provided for free
by the open source community.
They don't need the whole product,
and they probably never will.

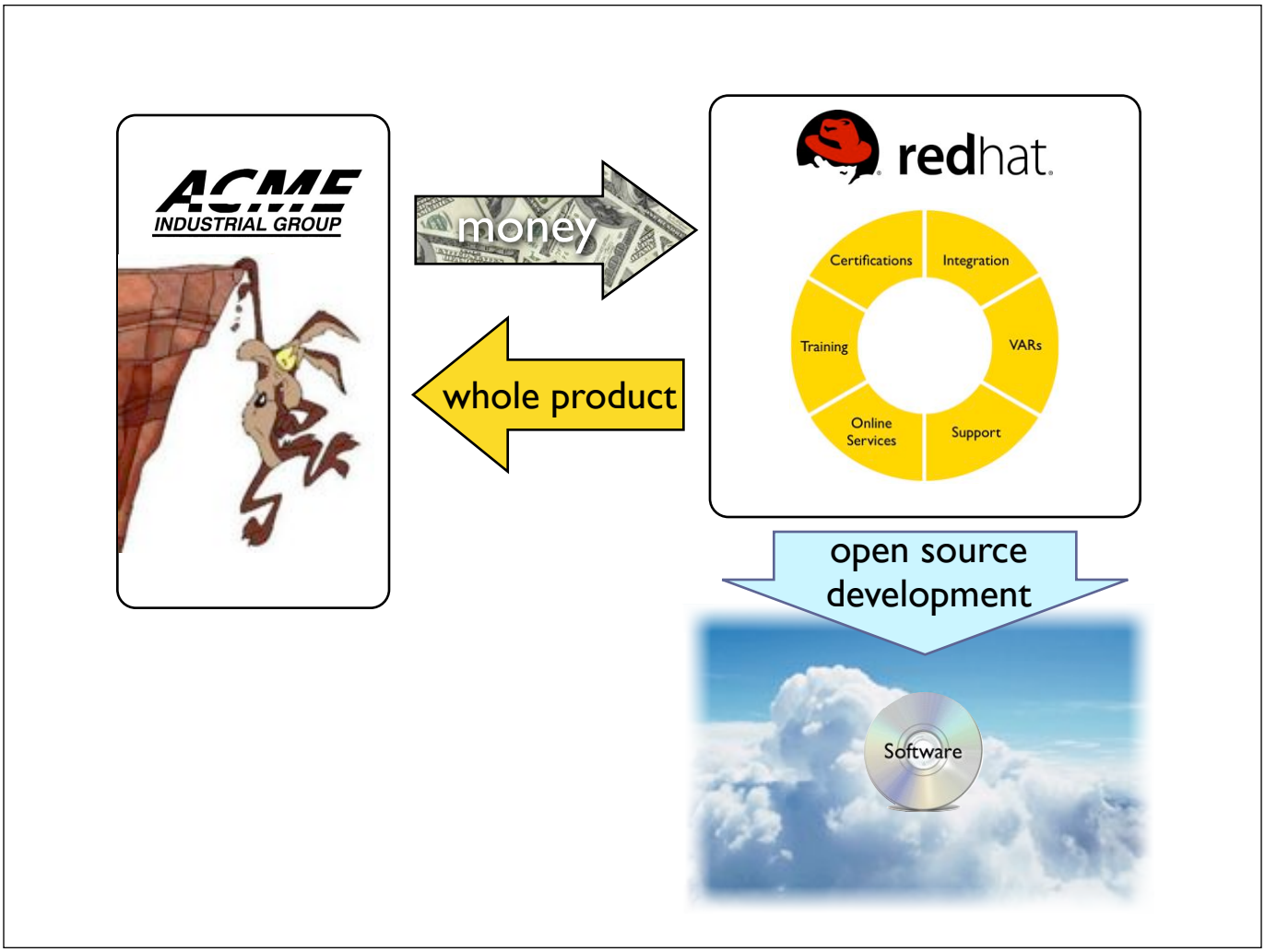


But what <X> about that
huge early majority and late majority?
What will it take to get
them to adopt open source software?
The low price of the software alone is
not sufficient to seal the deal,
because the
rest of the whole product
is missing.



<X> The long term open source business model,
as a general proposition, is about
providing a whole product
suitable for mainstream customers,
<X> but changing the point
of monetization.

Instead of companies selling software and
cross-subsidizing access to a
free network of services for customers,
we will have companies selling
access to a network of services and c
ross-subsidizing the development of free software.

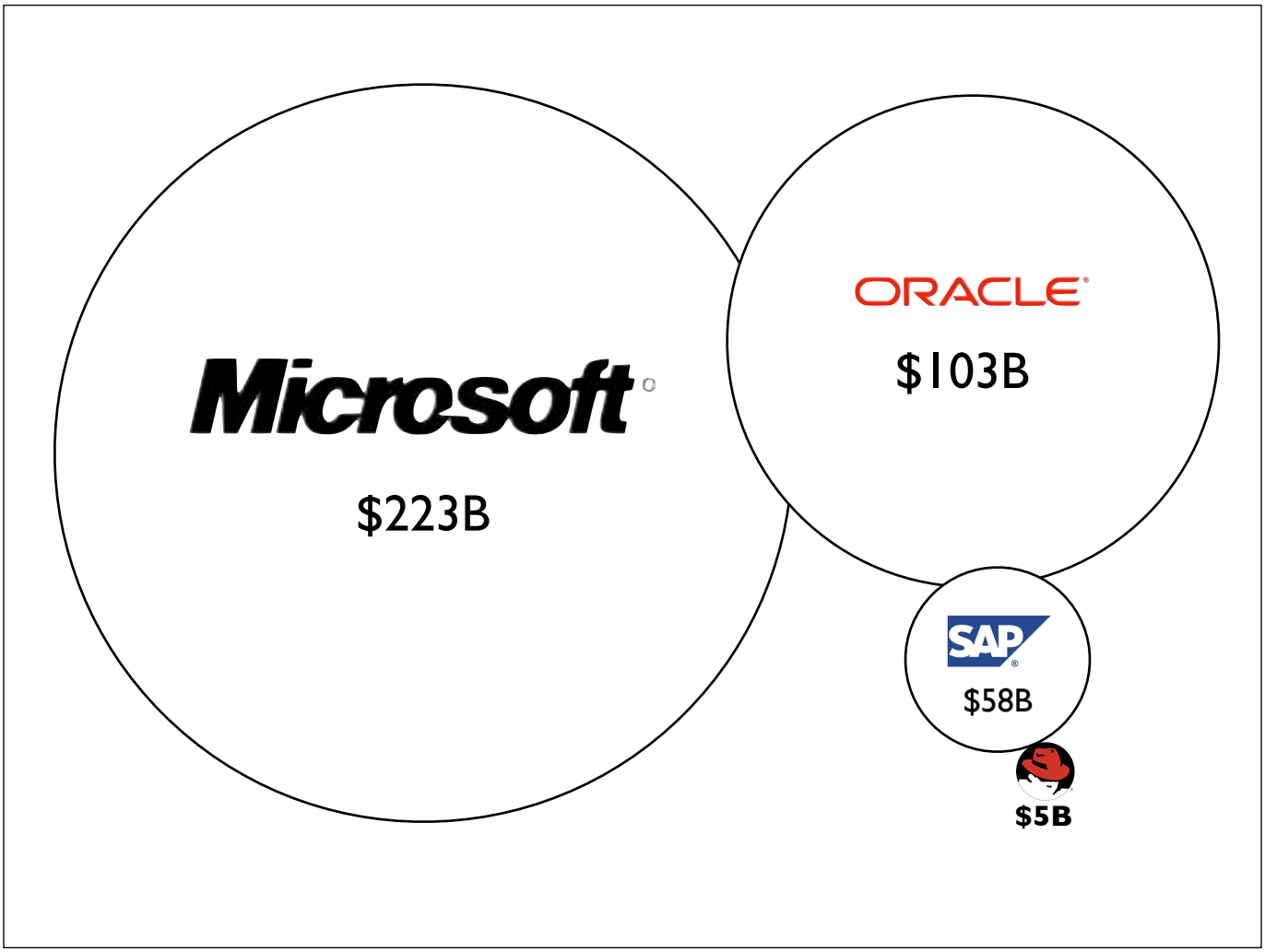


This is precisely the Red Hat Linux model.

<X> Say you've got some open source software,
being developed on the internet.
You've got Red Hat. And a risk-averse mainstream customer.
He gives <X> Red Hat money for Red Hat Enterprise Linux.
Red Hat uses that money to build a whole product
and to fund open source development.
They take a copy of the raw software,
wrap it in value added services,
and give it to the customer as a whole product.

The customer could simply <X> download the software directly,
but then he wouldn't get support,
automatic upgrades, testing, and so on.

<X> And for a mainstream customer, that feels like a risky situation to be in.



That's how an open source company is
supposed to work
in an ideal situation.

However,
it's not like the old proprietary model is fading away.
<X> Microsoft is a $223B company and
<X> Oracle is a 103B company and
<X> SAP is a $58B company.
The *biggest* open source vendor,
Red Hat, <X> is only a $5B company.

So, shouldn't I be worried?
Polishing my resume? <X>
Both open source
and open source companies
sound like helpless fluffy bunnies.
Cuddly and fun to play with,
but way overmatched in the marketplace.

Why am I so confident?

I'm confident because there is <X>
more to this story than
just the market for shrink-wrapped software. <X>
And there is a lot more
to the fluffy bunny than meets the eye.
The fluffy bunny is busy
transforming the field
of information technology
in profound ways,
leaving carnage in its wake.

May 28th 2009

"Open-source software has won the argument."

http://www.economist.com/opinion/displaystory.cfm?story_id=13740181

<X> For example,
the Economist magazine,
arbiter of free market orthodoxy,
has already taken in the situation and
declared open source a serious player:

<X> "Open-source software has won the argument."

May 28th 2009

"It is now generally accepted
that the future will involve
a blend of both proprietary
and open-source software."

http://www.economist.com/opinion/displaystory.cfm?story_id=13740181

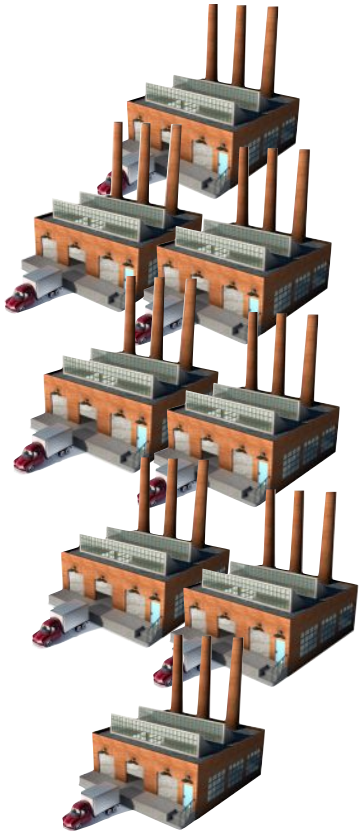
<X> "It is now generally accepted
that the future will involve
a blend of both proprietary
and open-source software."

How can this be?
The leading open source company
has a market capitalization less than
3% of the leading proprietary company.

How can open source "win the argument",
when it is so manifestly overmatched?

Here's how.

<X> First, understand that
you can't comprehend
the success of open source
exclusively by looking at the
marketplace.

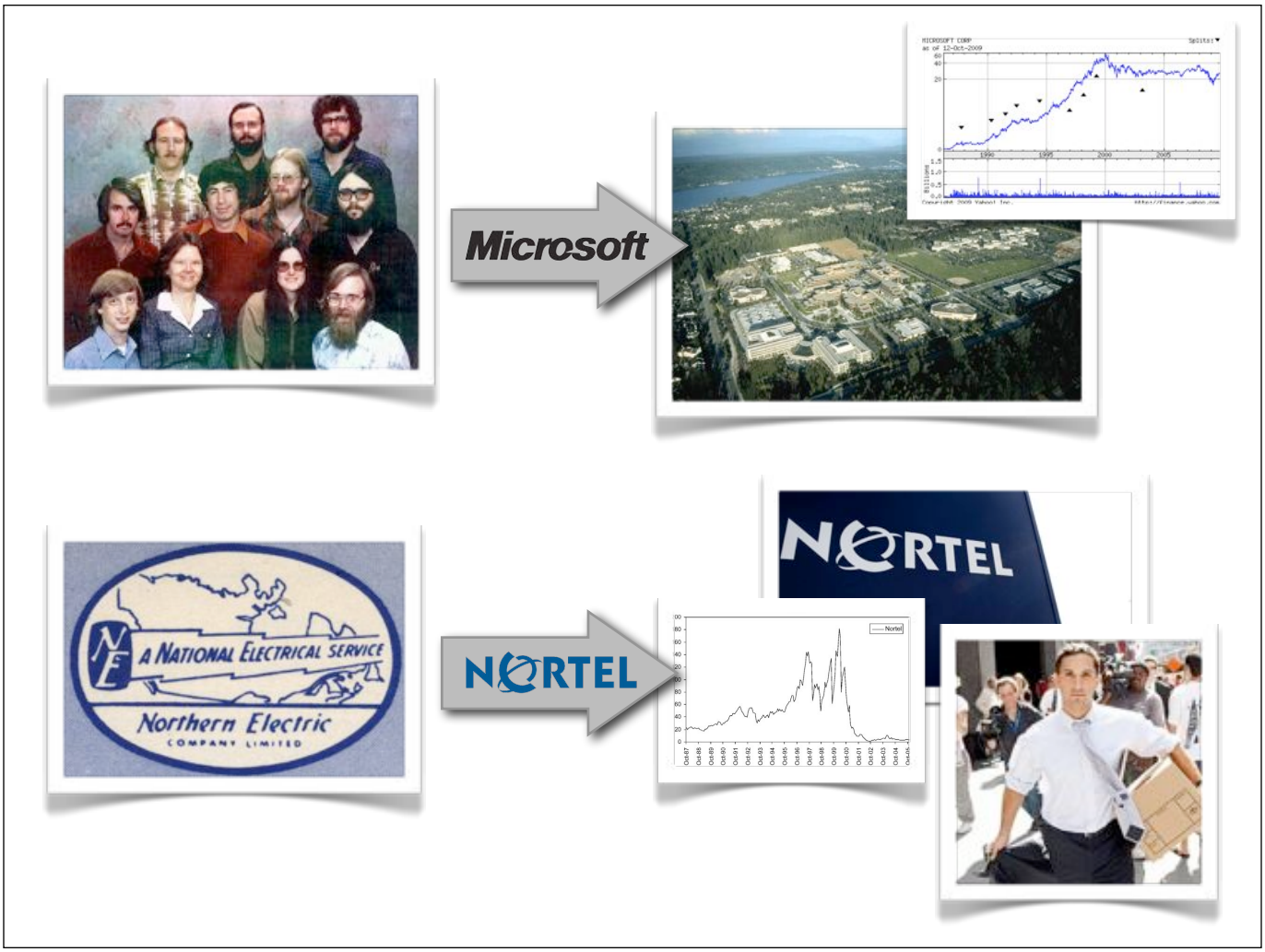


$REVENUE

\- $(EXPENSES)

$PROFIT

<X> In the marketplace
the unit of competition is a company,
and the measure of success is profit.



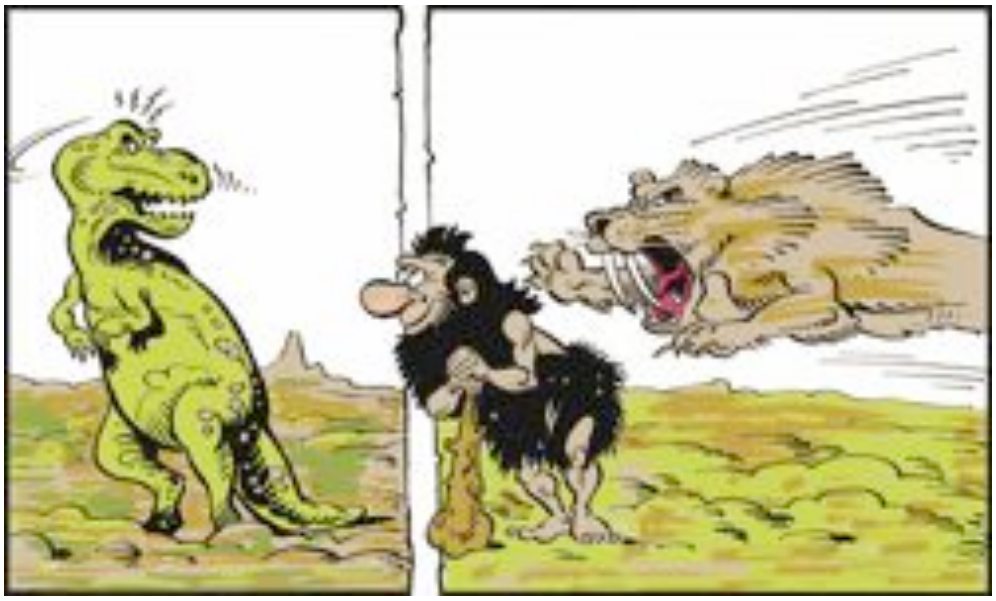
<X> The more dollars you take in,
the more successful you are,

and if you take in too few dollars <X>
you go extinct.

economy	ecology
companies	organisms
markets	environments
money	food / energy
bankrupt	extinct

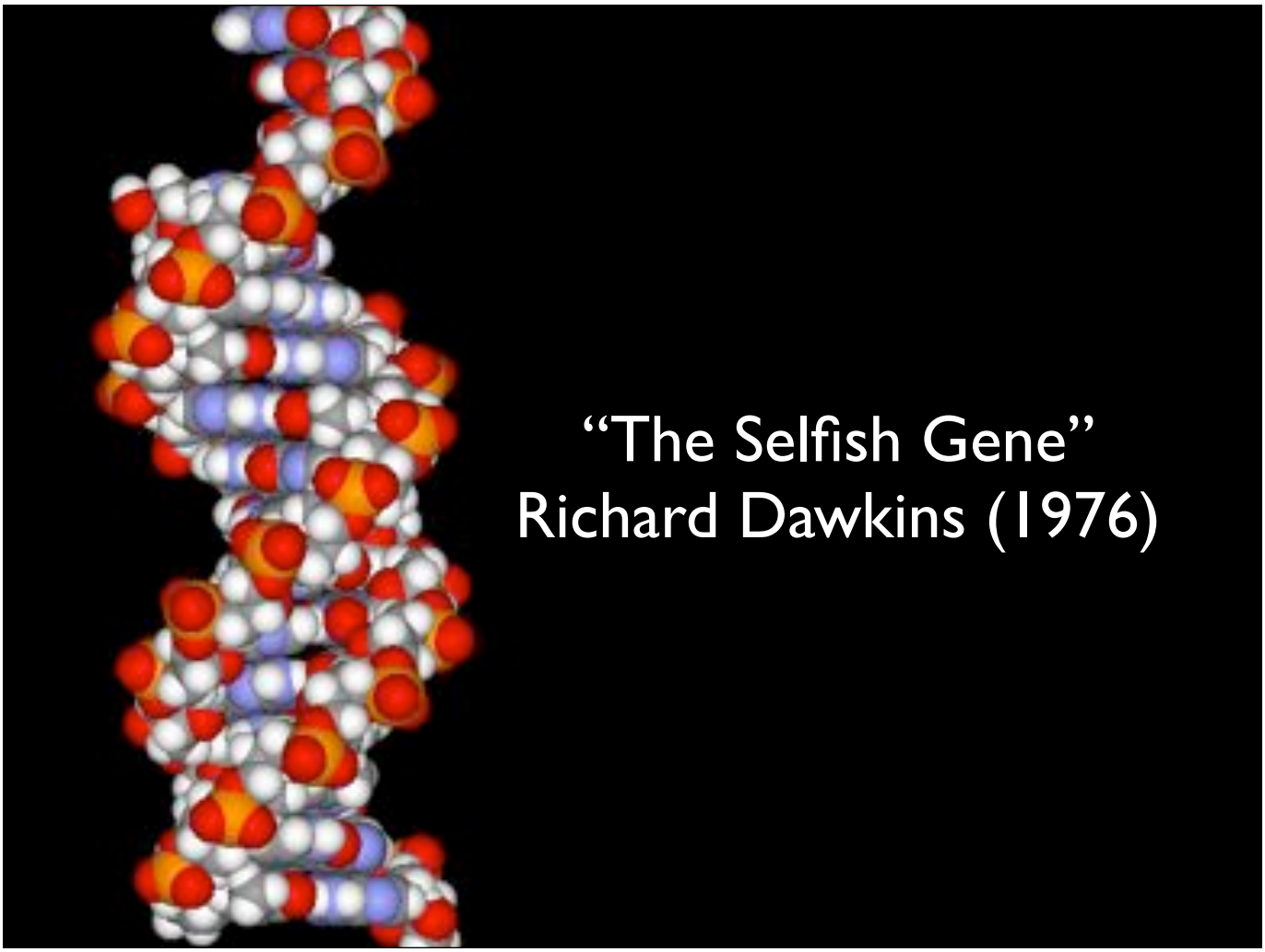
Lots of people have noticed that
there are parallels between the
market and the
field of evolutionary biology.
Taken to an extreme, you get
theories like social Darwinism,
but for for our case the metaphor
is instructive.
<X>

Only the strong survive.
<X> Those companies (organisms)
that best <X> adapt to their markets (environments)
<X> take in the most money (food / energy)
and those that do not
<X> go extinct (they end up bankrupt).



Also as with evolutionary biology,
it's <X> easy to be distracted
by the big lumbering beasts that
appear
to be directly engaged in competition.
And that's a bad thing!

<X> Because
the really *important* competition is
not at the level of the organism,
but at a lower level, much farther down,
in the realm
of the gene.



The competition between genes is
described by Richard Dawkins
in his book "The Selfish Gene",
<X> an exploration of how simple selfish
reproductive behaviour at the level of genes can lead
to apparently altruistic behaviours
at the level of the organism.

In Dawkins formulation,
behaviours that maximize the chances of
genetic survival are
passed to future generations,
even when those behaviours
endanger the survival of
a particular organism.

distraction display

Here's <X> a behaviour that makes
no sense in a organism-centric model.

This is a Killdeer on Vancouver Island, where I live,
engaging in a 'distraction display'.
The parent bird puts on an elaborate
(and totally fake)
display of being injured,
to draw an approaching predator
(or, in this case, videographer)
away from its young.

The parent organism is placing itself at great risk.
Why? Because the strategy is a good way to
preserve the genetic heritage of the
children in the nest.

The Selfish Gene

The trouble with these [other] books is that their authors got it **totally and utterly wrong**.

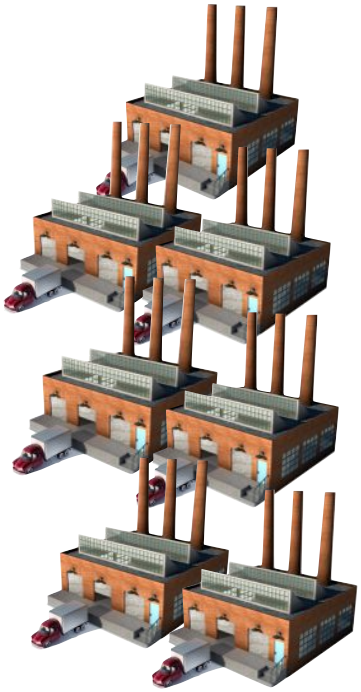
They got it wrong because they **misunderstood how evolution works**.

They made the erroneous assumption that the important thing in evolution is the good of the species (or the group) rather than the **good of the individual** (or the gene)

<X> "The Selfish Gene" came out in 1976,
and in the opening pages,
Dawkins has this to say about
some of his contemporaries,
(and you can see right away where he gets his
reputation as a gentle and self-effacing man),

<X> "The trouble with these [other] books
is that their authors got it
totally and utterly wrong.
<X> They got it wrong because they
misunderstood how evolution works.
<X> They made the erroneous assumption
that the important thing in evolution
is the good of the species (or the group)
rather than the good
of the individual (or the gene)"

corporation

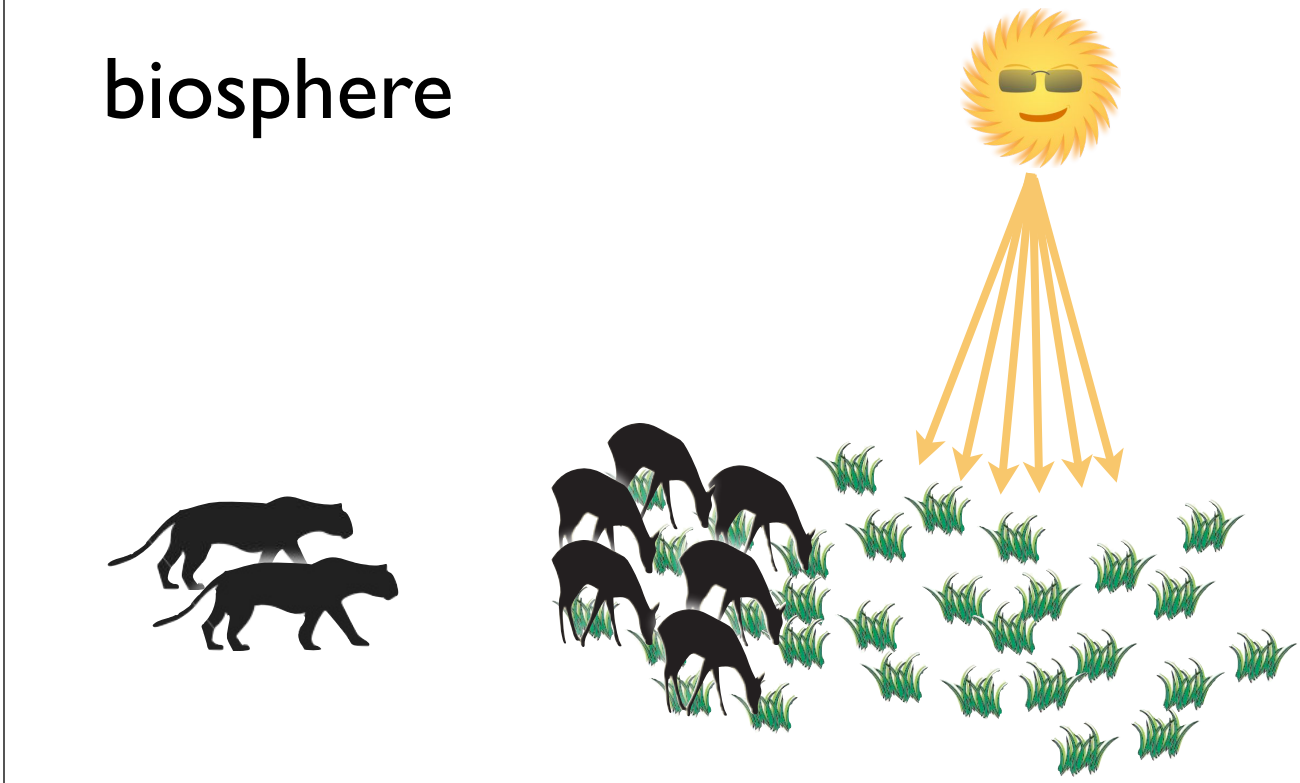


program

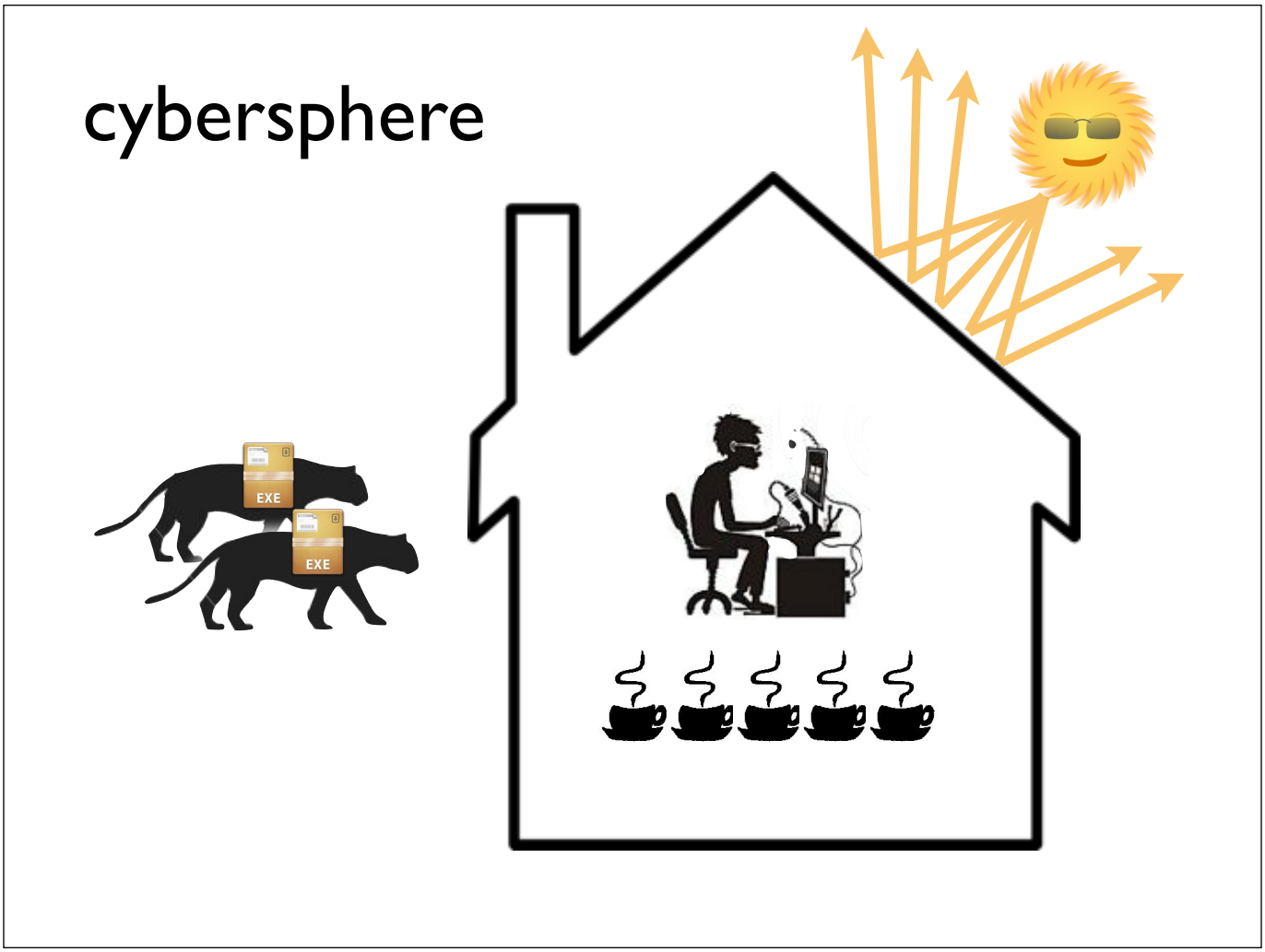
The same criticism
applies to people
who attempt to understand
the success of open source
through an analysis of the marketplace.

They misunderstand how software lives and dies,
confusing the host with what it carries.
<X> Because the unit of competition
in the world of software is
NOT the corporation,
<X> it is the PROGRAM,
it is software,
it is code.

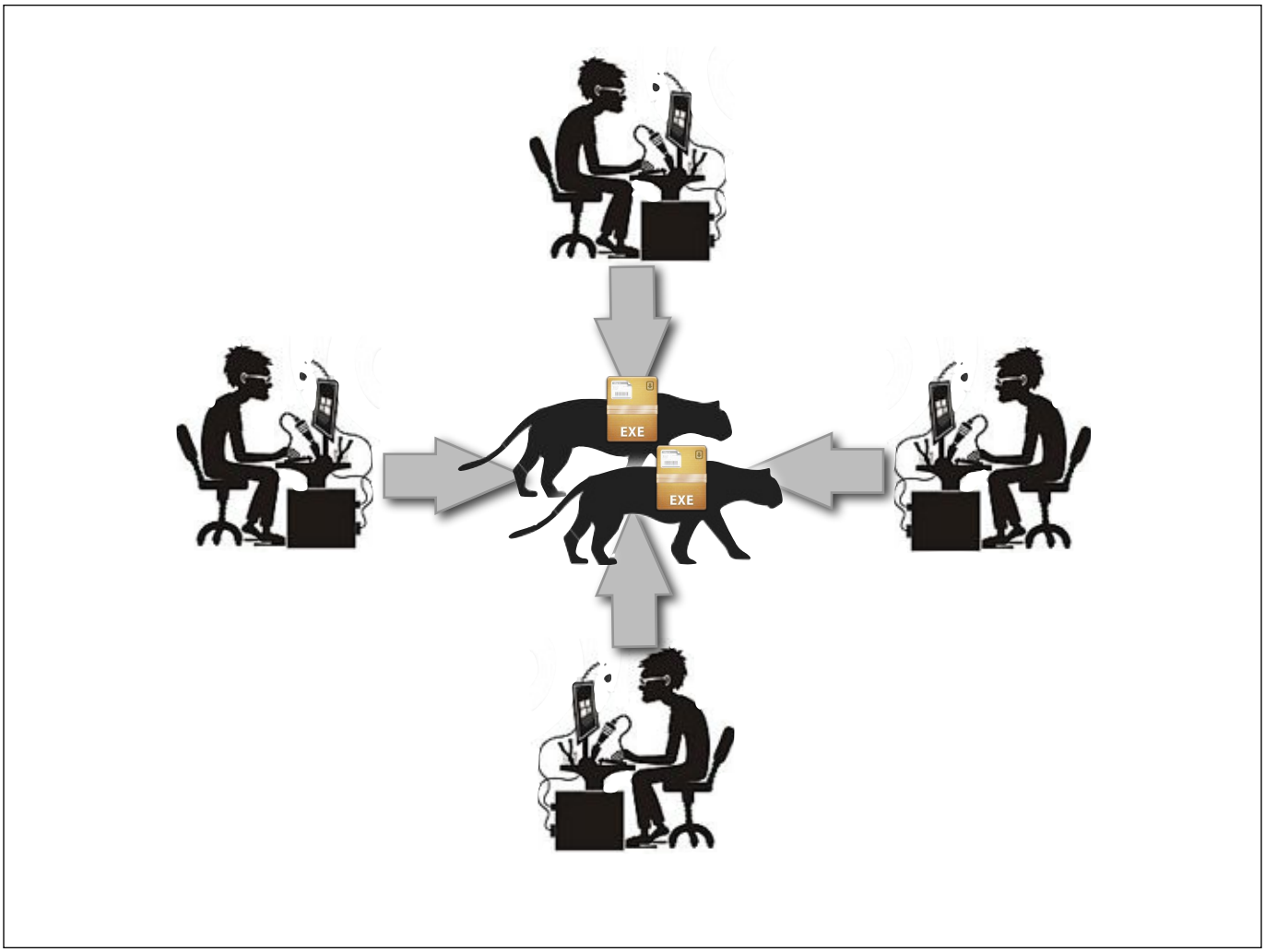
biosphere



<X> In the biosphere,
<X> organisms feed on other organisms,
which feed on plants,
which feed on light from Mr. Sun.
So, in the end the competition is for
sources of *energy*,
either direct (in the form of sunlight)
or stored (in the bodies of plants, and other animals).



<X> In the cybersphere,
<X> programs also compete for resources.
The resource that programs compete for
is developer time --
<X> commonly known as human attention.
So programs "feed" on developers,
<X> who in turn feed on caffeinated beverages
and try to stay away from Mr. Sun.



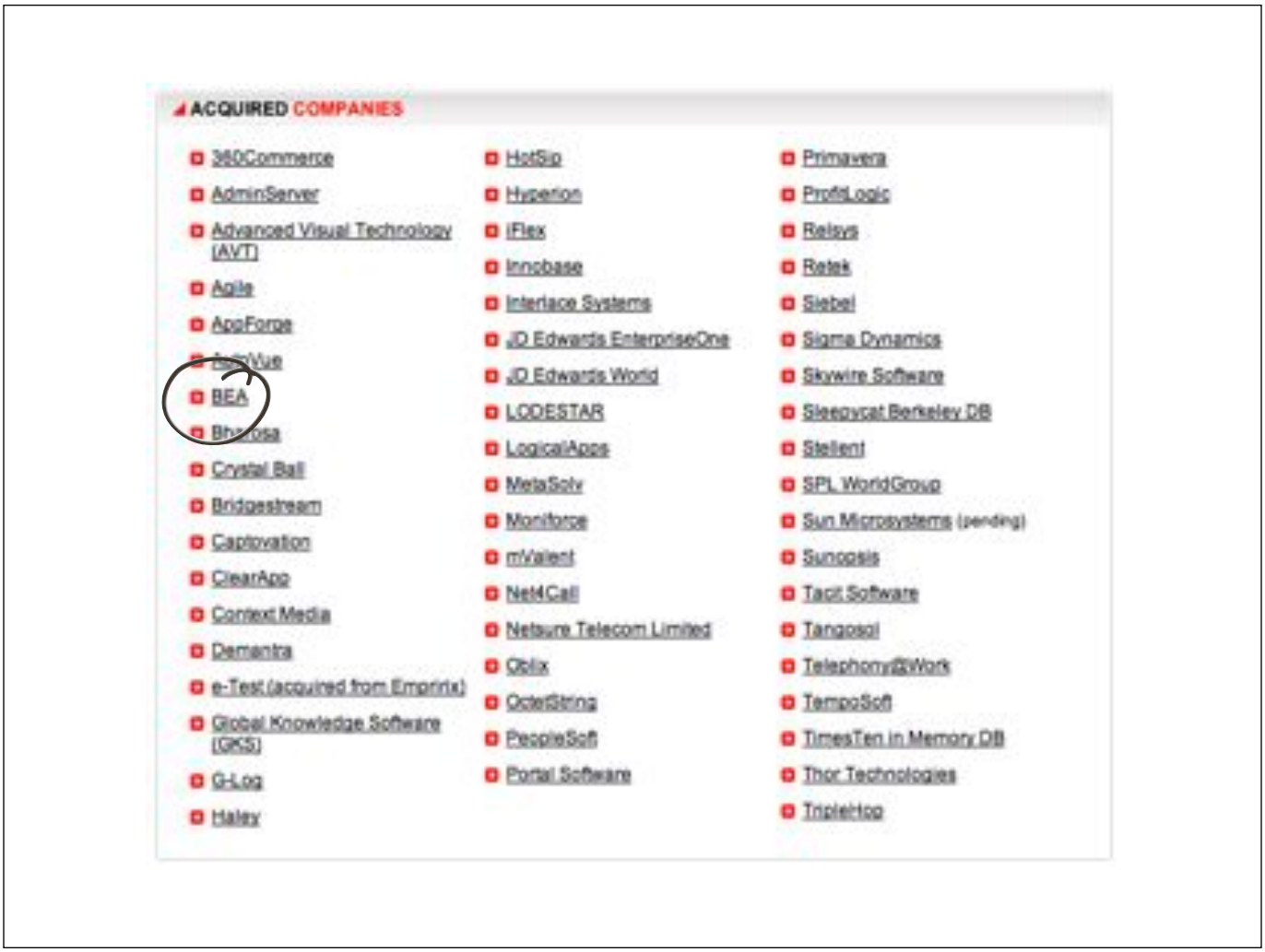
Programs need
programmer attention
<X> to survive.

A program that is no longer
being maintained and updated
is a program that is dying.

NCSA Mosaic

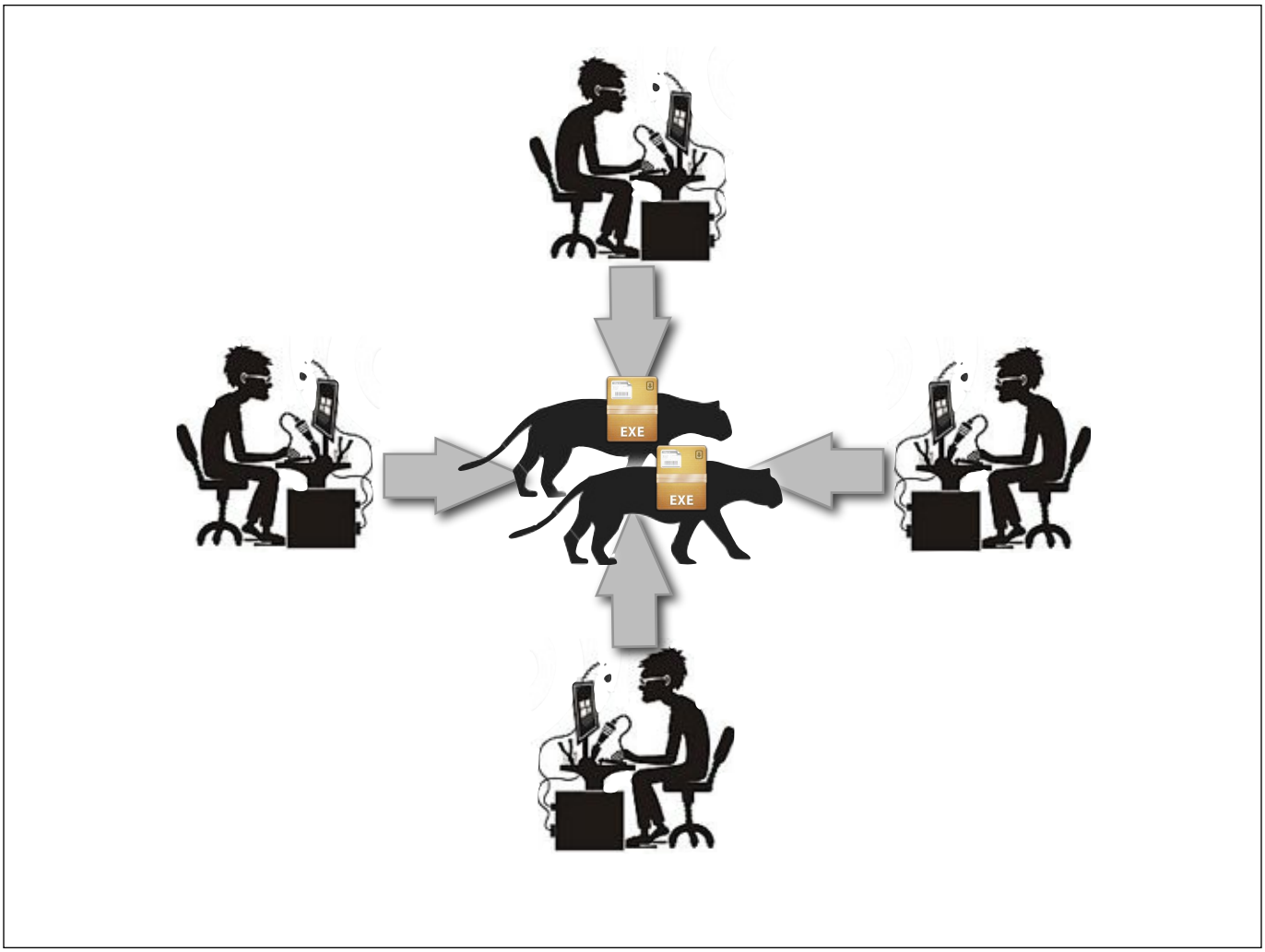
<X> A program that no one has a use for, is dead.
First it will be abandoned, un-run,
then it will become un-runnable,
and then it will be deleted.

Programs need programmer attention to survive.



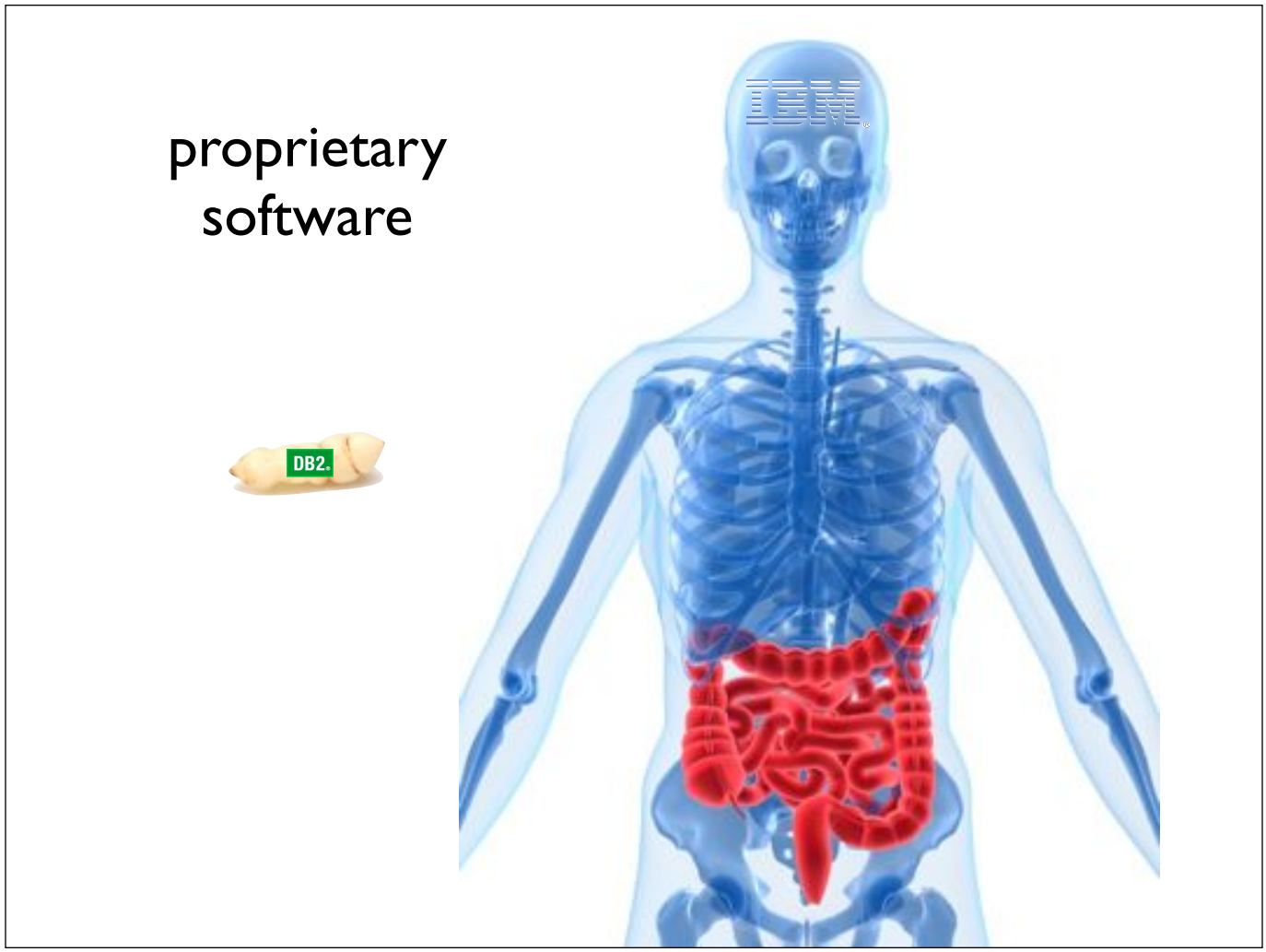
Don't believe me?
<X> When Oracle gobbles up
yet another enterprise software company,
do the customers bemoan the death of the *company*? <X>

No, primarily they worry about the *software*:
Will bugs be fixed?
Will we get that next release with the new features?
Will the developers flee to greener pastures?
<X> *Will the new owner continue to feed the software?*

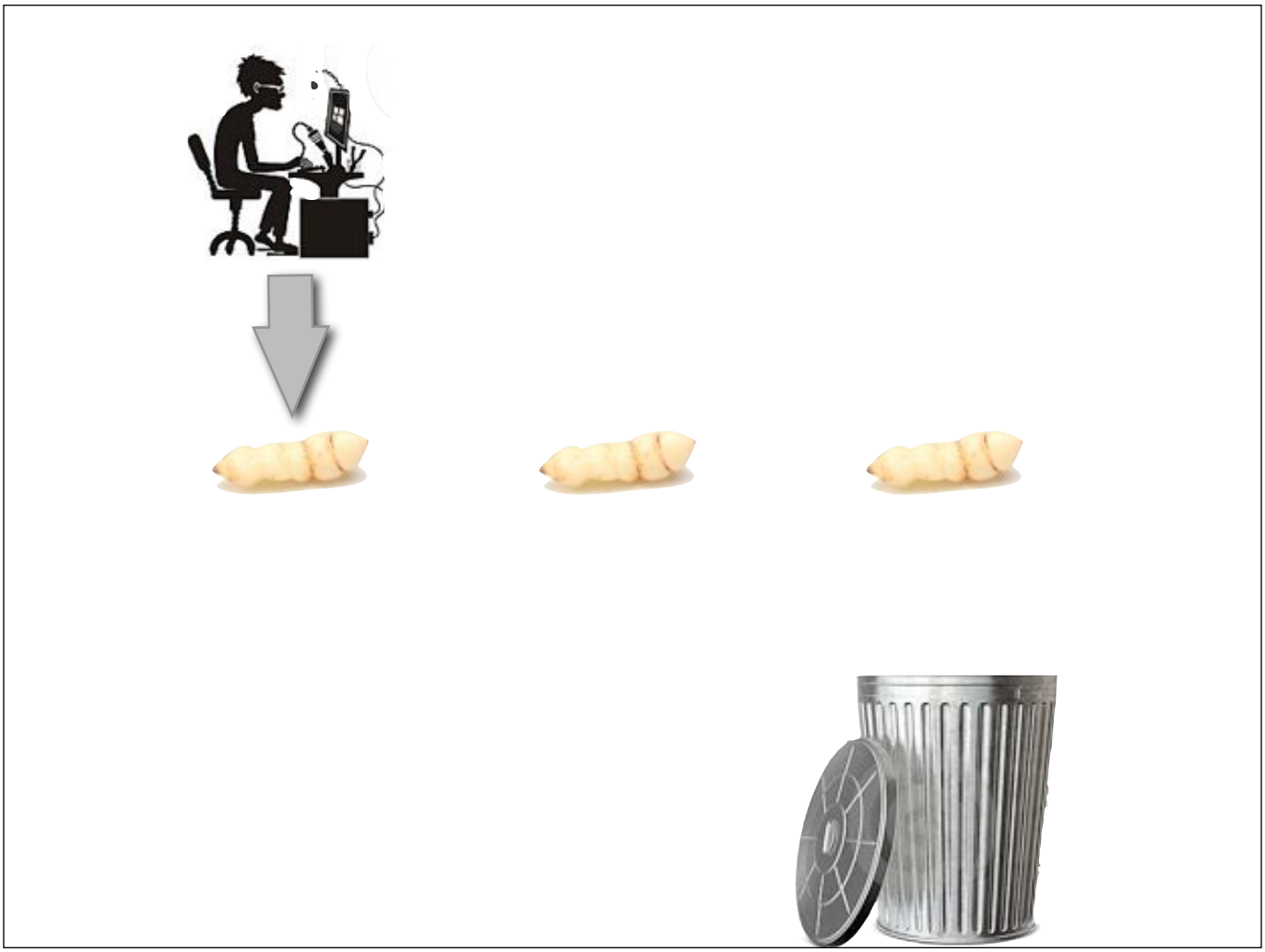


And what does the software worry about?
All that matters to the software is that it
continues to receive a steady supply of developer time.

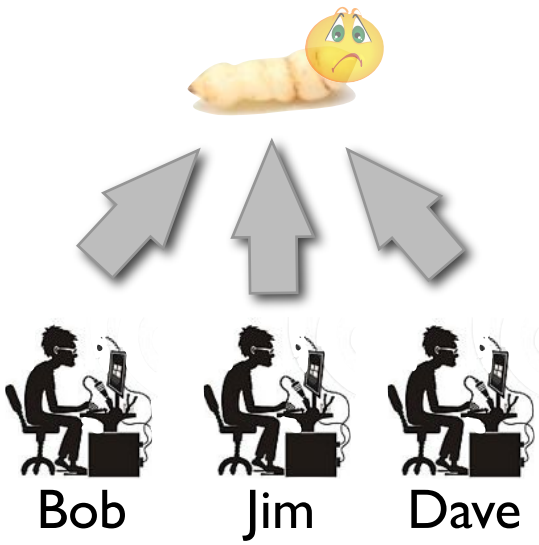
Understanding the competition between
proprietary and open source
as a competition at the
program level
clears away a lot of distractions.
Now, we can directly evaluate
which strategy is the most
adaptable strategy for survival:
the open source model;
or the proprietary model?



A proprietary program
can best be understood <X>
as a form of parasite.
It resides in symbiosis <X> with a host organism,
the corporation that owns it,
and draws its sustenance
exclusively from the
developers provided by the corporation.
The amount of sustenance provided to the
program pretty directly correlates
to the success of the corporation selling the program
(though sales success may or may not
correlate with the actual quality of the program).
When the corporation dies,
the program usually dies too.



If the corporation is subsumed by another corporation,
<X> the new host may continue to feed the program,
starve it to death,
or terminate it immediately
in favour of some other program.



When you think about it that way,
it is easy to feel a little sorry <X>
for a proprietary program.
It is very much at the mercy of its host.
Its success or failure may have
nothing to do with its intrinsic quality.
It may have <X> only a small team of developers to love it,
feed it, and carry its memory forward if it should die.

In contrast to the sheltered,
monastic existence of proprietary software,
<X> the lifestyle of a successful
open source program is
incredibly promiscuous.

Any developer with a nice smile and
a good patch is welcome to join the party. <X>

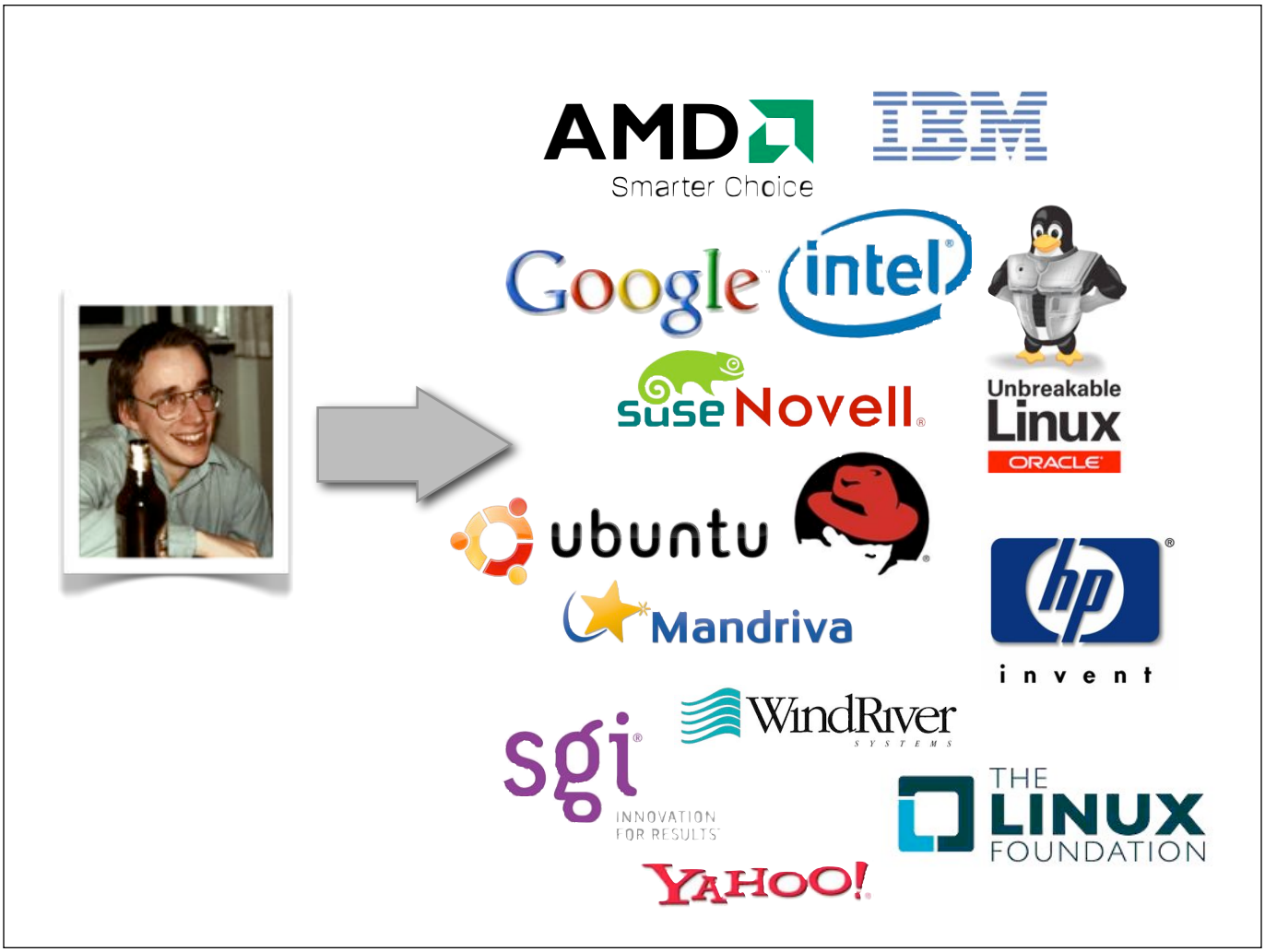
Open source programs can draw sustenance
in the form of long term,
stable commitments from corporations
who sell services or products around the software,
<X> from devoted contractors
who derive income from contracts
for features development or bug fixes,

or from <X> quick relationships with casual developers
who just drop off a patch and run away.

In contrast, proprietary programs are
embedded within the <X> institutional framework
of a corporation,
so it is much harder for them
to form relationships with
new sources of development --
people can't just stroll in the door a
and add a new feature to Microsoft Word.

<X> The relationship between developer and proprietary software is formal, contractual and exclusive.

Open source programs
can form relationships with
<X> multiple developers and
multiple organizations simultaneously,
because open source is
not trapped inside a single organization.
The rules of participation are
cultural, not contractual,
and broad community participation is
the WHOLE POINT.



Take the most successful open source example, <X> Linux.
The Linux software has gone
from drawing development effort from
a single Finnish graduate student,
<X> to receiving the attentions of
hundreds of fully funded developers
in multiple fortune 500 corporations,
government agencies and
academic institutions.

Even organizations that are in
direct market competition --
IBM and Oracle, or
Red Hat and Novell --
provide code for Linux,
as do thousands of other developers with
institutional affiliations ranging from
top secret government agencies
to academia,

cat

coffee vendor

to individual developers
whose only real affiliation <X> is to
their cat and
their coffee vendor.

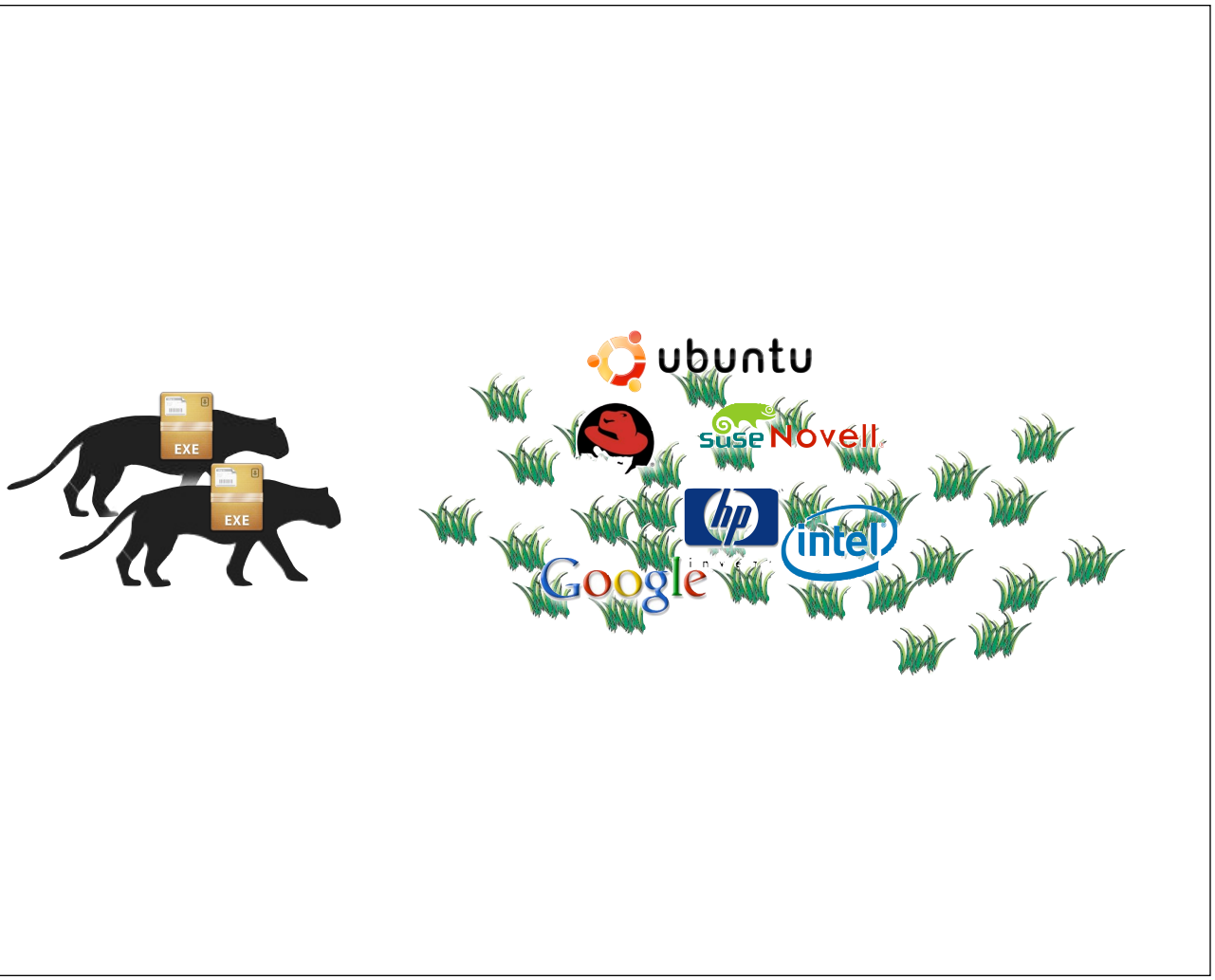
unaffiliated

17%

of the linux kernel

http://www.kroah.com/log/linux/lpc_2008_keynote.html

In fact,
kernel developer Greg Kroah-Hartman
did a study of the kernel source code
in 2008 and found that
the number one developer affiliation <X>
was "unaffiliated",
accounting for 17% of kernel.
Red Hat was #2 at 11%.



<X> The example of Linux
shows that open source programs
<X> are not limited to feeding off of
pure open source companies.
They can feed off of
<X> any company that
derives competitive advantage
from using open source.

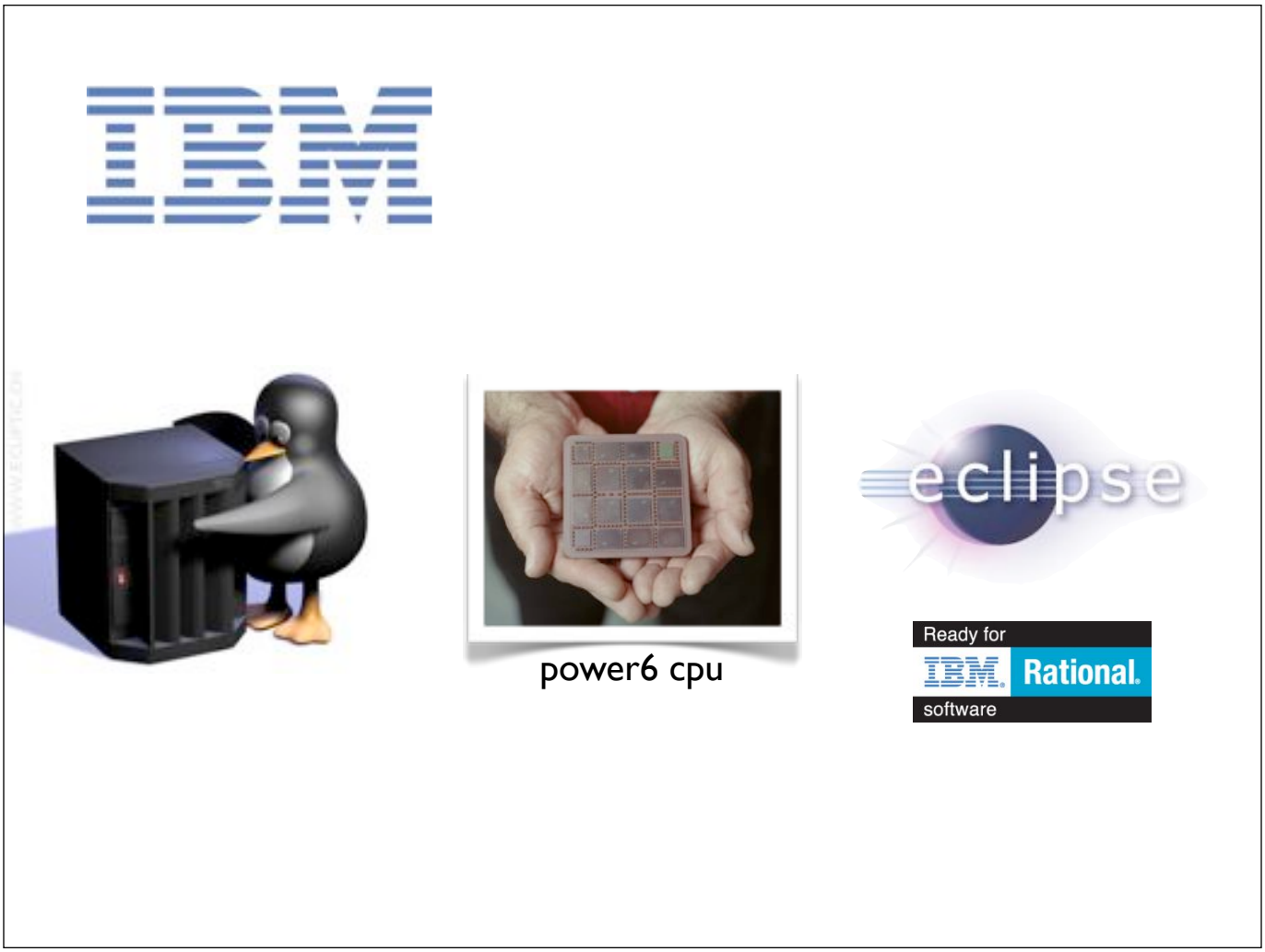
Matt Asay

"We are all open-source companies now.
Which also means that none of us are."

http://news.cnet.com/8301-13505_3-10354530-16.html

One of the most perceptive
observers and commentators
on the entrance of open source ideas
into the marketplace <X>
is Matt Asay.
He recently wrote
<X> "We are all open-source companies now.
Which also means that none of us are."

What he means
is that every company in the marketplace
is now deriving competitive advantage
from open source in one way or another,
even deeply,
deeply proprietary companies.



<X> IBM was once so proprietary that
Microsoft looked open by comparison.
But in 2000, <X> IBM was the
first major company to adopt a Linux strategy.
They invest directly
in Linux kernel development
to ensure it runs on <X> their CPUs and systems.
They are a founder
<X> of the Eclipse Java framework project,
and a number of their proprietary products,
like Rational <X>, are built on top of Eclipse libraries.

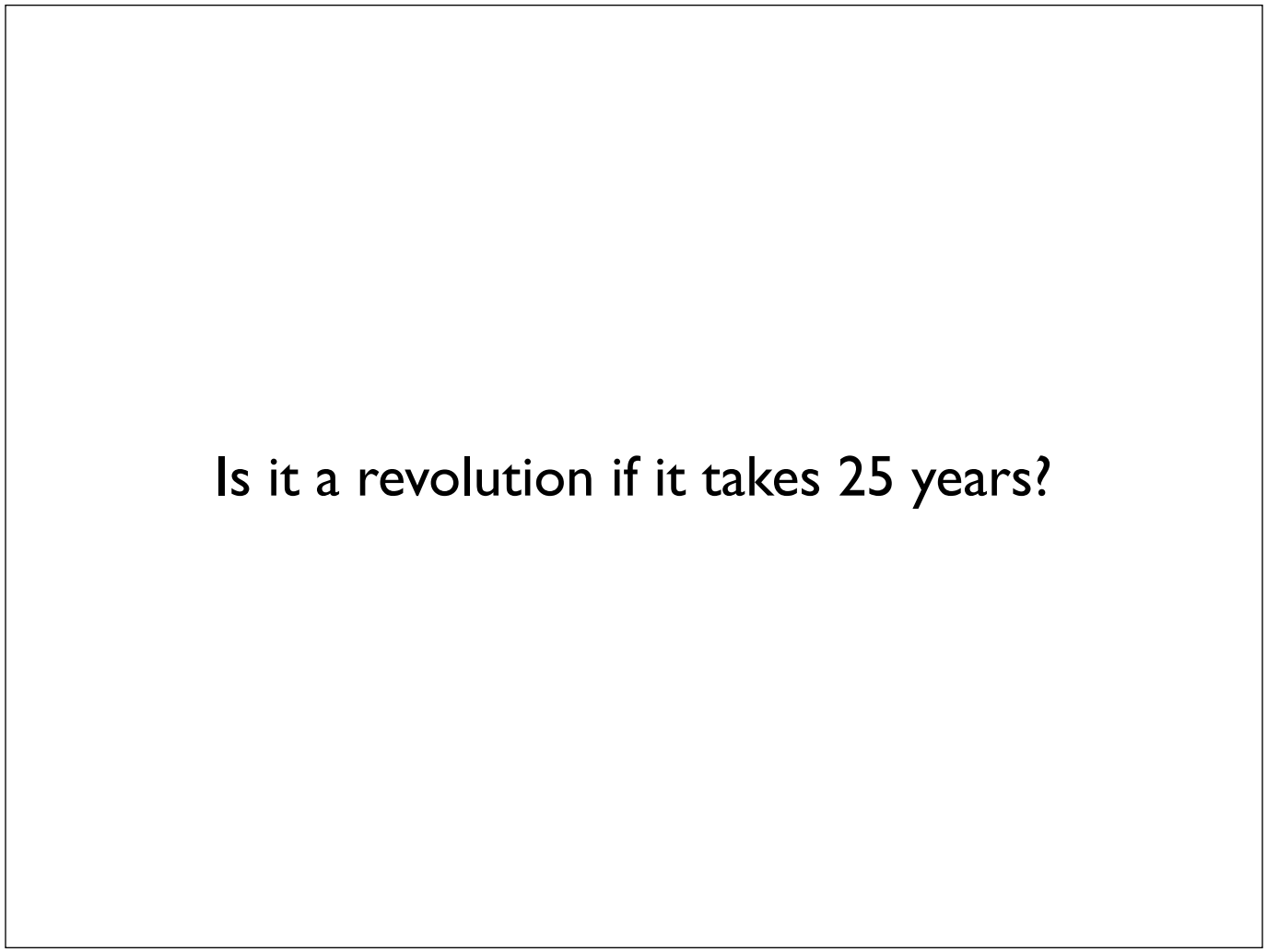
<X> Oracle
has purchased
several open source companies,
over the last few years,
<X> database companies InnoBase <X> and Sleepycat,
and this year purchased Sun, <X>
which netted them some
very <X> well-known open source names.
And they aren't just sitting on them.
At Oracle OpenWorld last week,
Oracle's evil genius <X> promised to
invest even more money in MySQL R&D than
Sun is currently spending.

Even <X> Microsoft,
which practically invented the idea of
proprietary shrink-wrap software back in the 70s,
now has an active open source strategy.
They have an open source
code hosting repository, <X> CodePlex.
They are a sponsor of the <X> Apache Foundation.
They invest in the development of
Windows-compatible open source,
<X> like IronPython and now <X> even PHP.
They have even <X> contributed patches to the
Linux kernel
under the GNU GPL.

In our own industry too,
the momentum is towards
more and more <X> open source use.
ESRI uses the GDAL raster library in ArcExplorer.
So does Google Earth.
<X> PostGIS is becoming an
industry standard spatial database,
supported even by old guard companies
like ESRI and MapInfo.

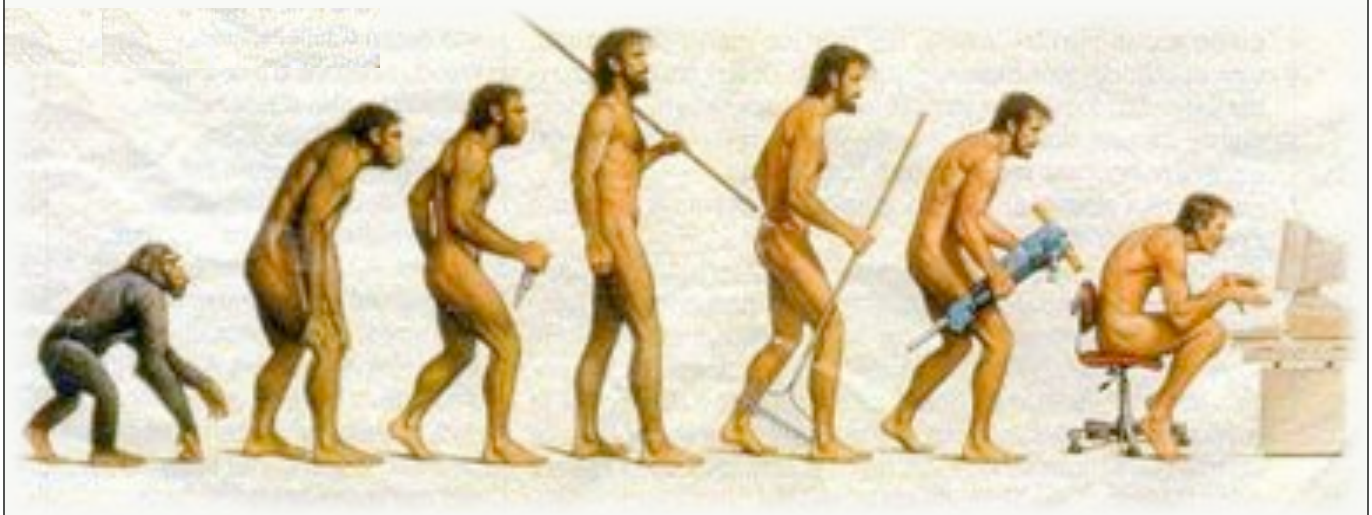
When even
the proprietary companies are
investing in open source,
what does it mean to be an
"open source company"?
Everyone is doing it!

People like to talk
about the change from
proprietary <X> to open source as an
"open source revolution".
But revolutions are quick, turbulent affairs <X>



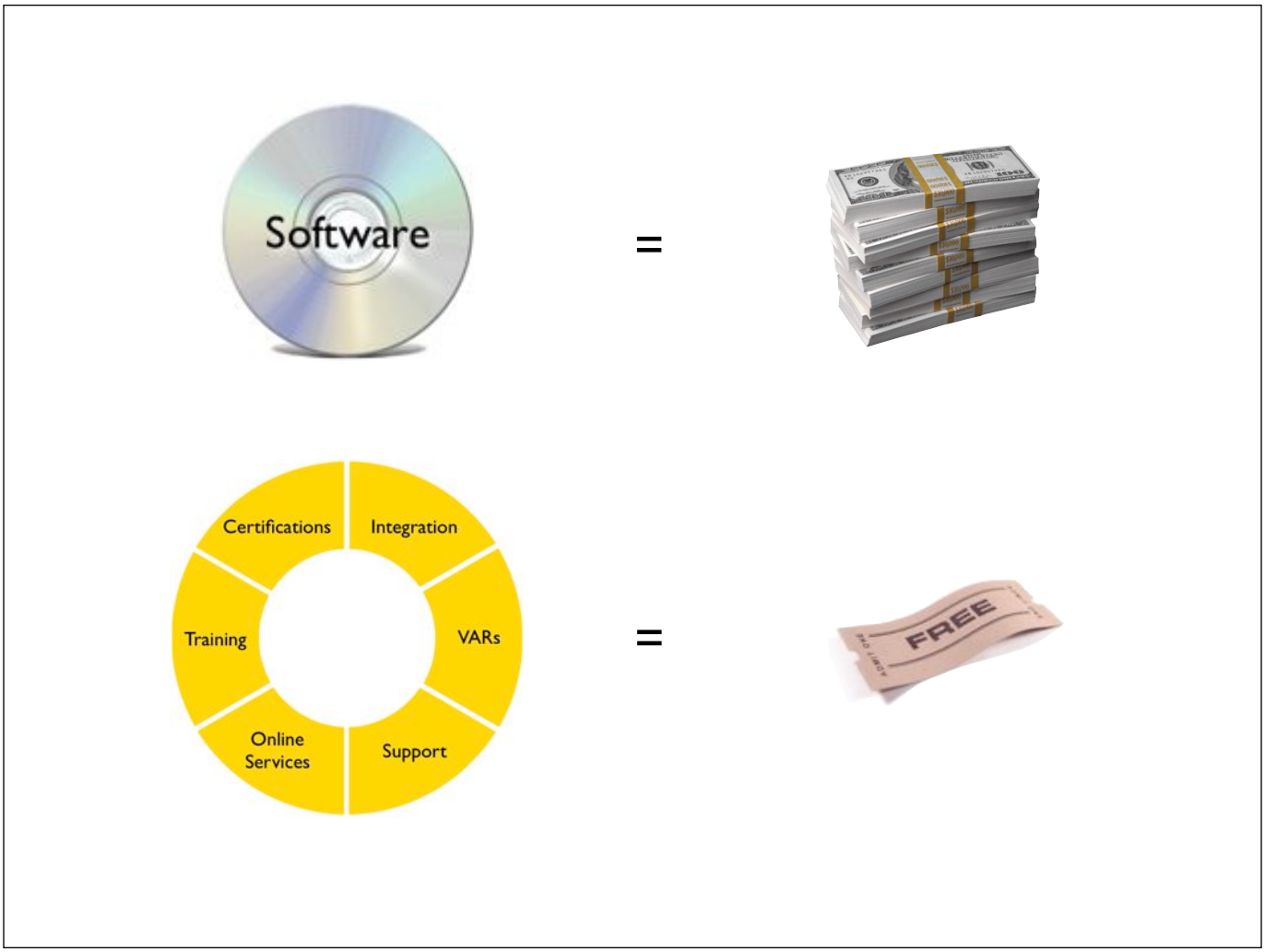
Is it a revolution if it takes 25 years?

open source ~~r~~evolution



I think that what we are experiencing
is not an "open source revolution",
<X> it's an "open source evolution".

The progress is slow and incremental, but
the movement
is always in the same direction,
month by month, and
year by year.



<X> We are just at the start
of a transformation in the software market,
<X> where purchasers recognize
that they have the option to buy the whole product
and get the software for free.

And we are in the middle
of a transformation in how we build software,
moving very quickly <X>
from a closed corporate model,
 where source code is private;

<X> to an open collaborative model,
where source code is a commons.

And it is the
combination of those two trends
that
fills me with confidence.
Because the two trends
are re-enforcing each other.

And that's why
I can look my mother-in-law in the eye and say
"don't worry, it'll all work out".

<X> I'm on the side of history,
on the ground floor or a growing market,
riding a wave that is just picking up.

And so are all of you.
Let's make the most of it! <X>

Thank you.