1

Connected Services: The Collision of Internet with Telco


```
For coolnames.each do |c|
display c.coolword
call ubiquity
End
```

- Any digital service that brings people together in a meaningful way – to engage, transact, share, and so on – is a "connected service". This includes digital communications in telcos and on the Web.
- The real backbone of connected services is software, not networks. The dial tone of connected services is "http://".
- A common architectural pattern for connected services on the Web is open platforms. Successful platforms enable digital ecosystems to flourish.
- Using standard telco business models to explore the value of a service isn't congruent with the value of Web platform services.
- When building a platform, user experience remains important.

1.1 Connected What?

An uninformed observer, or visitor from a distant galaxy, would be forgiven for thinking that telcos ought to have been at the heart of the Internet revolution that has swept through much of the developed world these past 10 years, following the tipping point of the Web. After all, telco is all about networks, as is the Web. Telco is all about connecting people, as is. . .

You've guessed it – The Web!

Instead of Google, Yahoo, Facebook, Flickr, our alien visitor might expect to see the icons of the Web to be O2, Verizon, Orange. . .

But they won't. Figure 1.1 shows what they will see (you will probably recognize most of them):

Connected Services: A Guide to the Internet Technologies Shaping the Future of Mobile Services and Operators, First Edition. Paul Golding.

Figure 1.1 Web logos – spot the telco!

Users surfing the "mobile web" often arrive at their digital destination via the on-ramp of Google search. Users finding their way across town often arrive at their physical location via Google Maps. Developers are hacking with Google's Android. Users finding old friends, and making new ones, are doing so via Facebook. Business folk are connecting and networking via LinkedIn. And on the list goes, dominated by companies that all appear to have one thing in common – they were born on the Web.

But don't they have something else in common? That's right. They all appear to be obsessed with *connecting* people, to other people, to data, to places, to whatever – to things? Furthermore, many of these ventures were born in universities, although often not in the labs. They were born in the dorms and sometimes in the coffee houses. This is a key symbol of the Web 2.0's innovation *culture*. There you go – I did it – I used a dreaded "C" word.[1] And I mean *culture*, not connecting.

Reluctant to use that particular "C" word as I am, as it generally sends corporate minds in a spin ("What is culture?" "How do we change culture?"), I am not going to shy away from talking about "non-tech" stuff in this book wherever it serves to make a valuable point. You see, in my experience, technological enterprise – the art and science of really getting something done, something worth doing with tech – is not done in isolation of people, attitudes and verve. This point, perhaps more than any other, might explain why the Internet is not dominated

[1] Doesn't every book related to telco have to have a list of words beginning with C? The five Cs? The three Cs? Not sure what the magic number is these days.

by telcos. They are different types of enterprise creature, if you will, or should I say ecosystem (more on that later).

It's not as if telcos didn't have the money to build substantial Web ventures. Well, some of them tried, and failed. It's not as if they didn't have lots of "technical people" either, or, more importantly, lots of paying people who make up those incredibly large customer bases that would be the envy of any Web start-up and most Web ventures. Maybe they didn't have the right cultural conditions. They mostly still don't. And the only reason I mention this now is that Web 2.0 is as much about culture – the way people think and behave *by habit* – as it is about technology and business patterns. If you work in a telco and you still don't get this point, then I recommend reading this book on an airplane, one destined to Silicon Valley where you can hang out with Web ventures and see how they really work.[2]

Sure, 99 per cent of this book is going to be about tech stuff, but that's almost irrelevant if you don't set up the conditions to make the tech work for you. I know what I'm talking about. My first book – *Next Generation Wireless Applications* – explained much of the Web 1.0 tech, the emergent Web 2.0 stuff, and its mobile offshoots in great detail. That was back in 2004 (and I started writing in late 2002). I wrote the follow up in 2007/8 sprinkled liberally with 2.0-isms.

Both these books were bought mostly by folk in the telco ecosystem – and then mostly ignored. I know, because I held numerous workshops based on the books' themes. I got the feedback firsthand, which was almost always a room full of "Why would we do that?" and other "Why?" questions that added up to a unanimous "We don't get this . . ." message, which is cultural, not technical. Culture is embedded in language and, if you don't speak the lingo, you really won't get the culture, not in any depth.

I set up one of the first mobile ISVs in Europe, back in 1997. I built the first Mobile Portal ever (Zingo) back in 1998, which we (i.e. with my client Lucent Technologies) took to Netscape as their mobile play – imagine that, a telco supplier (Lucent) pushing product to a Web darling. Whilst acting as Motorola's Chief Applications Architect 2005–7, I set up their "Mashing Room" lab to build hacks that would demonstrate the intersection of mobile and Web 2.0 – "Mobile 2.0," if you will. We built a telephony mash-up not too dissimilar to Google Voice (previously Grand Central). I spent much of those two years evangelizing various Mobile 2.0 themes to operators globally. Again, my enthusiasm and ideas were mostly met with blank stares.

Which brings me to the next "C" word – COLLISION!

That's pretty much what's happened. Web 2.0 has hit the telco world, almost taking them by surprise, even though it's been a gradual creeping up, like the vine that slowly grapples a wall (and pulls it down). The overwhelming sentiment is that "these guys" – that is, the Web companies – are slowly eating our lunch, and they're doing it using our networks (bit pipes). What's more, they appear to be doing what we do, don't they? Connecting people!

No need to debate this point. Let's get straight to the killer question:

"What can be done about it?"

This brings me to the final "C" word of the series (noting my dear readers that every seasoned evangelist has to tell a story using 3 or 5 Cs at least once in their career):

"*CONNECTED services!*"

[2] This is not a flippant point. I took one senior telco guy on such a trip and he came back "converted."

This phrase happens to be one I've heard used by O2, one of the companies I consulted for when I was writing this book. But they're not unique in their ambition, which is to become something other than just a "mobile company" in order to avoid the inevitable descent to dumb bit-pipe, should they, or any other telco, not want to end up there, which is debatable (see Section 1.3 Six Models for Potential Operator Futures).

The phrase "Connected Services," is supposed to cast a wide net, and one that frees us from the constraints of a telephony network. Any digital service that brings people together in any meaningful way – to engage, transact, share, and so on – is a *connected service*. In that way, Twitter is a connected service. Facebook is a connected service. Even search is a connected service. I don't want to get too prissy about definitions, as experience has taught me that such distractions are exactly that – distractions. A quick skim of the contents page will tell you the sorts of stuff I mean by connected services. The issue for operators is that telephony is a very old technology that hasn't changed much. And, while people will always want to talk, at least for the foreseeable future, we can see that more and more people are finding ways to connect without voice, like the examples just given, which all take place on a giant platform called Web 2.0, quite separate from telco networks, which just carry the traffic to and from these various Web platforms.

So, what can be done about it?

This isn't one of those "get rich quick" books. There's no easy answer. . . .

Actually, there is, which is to do something different from what you've been doing. That's the easy answer, incomplete as it is. Nonetheless, many operators remain in limbo, trying to gain the freedom to innovate that evades them and blesses the innovators at the extreme ends of the "freedom to innovate" spectrum – the cash-rich Googles and VC-funded companies at one end and the cash-starved boot-strapping bedroom start-ups at the other.

As I keep telling my colleagues in the industry: "Think, try, fail, tune, deliver . . ."

You've got to stop pondering about all this stuff, stop thinking about a "them (Web) and us (telco)," and start building stuff, putting it out there and tuning as you go. This is the agile way, the Web way (see How Chapter 10). The battle-hardened roadmap process for deploying and running vast arrays of network infrastructure, supporting millions of customers and running giant marketing campaigns serves very little purpose on the frontiers of the Web. It doesn't matter if you're a 100-year-old company that dug up roads to wire the nation, when it comes to the Web, you're a start-up – it's still a frontier world where more is still unknown than known and where we continue to be surprised by the rampant success of "new" ideas (like Twitter) and emergent categories (like Social Networking). In this regard, most Web ventures are still start-ups, whether launched in a dorm or from the labs of the 100-year-old giant. And in the world of start-ups operating in the unknown, agility is king, as are other memes, as the Web-geeks call them, like platforms, real-time and "Big Data," all of which we shall explore in enough tantalizing detail to get you motivated to try something different.

This book is about the ingredients, patterns and technologies that will enable connected services to work in the Web that's emerging post Web 2.0. Is that Web 3.0? Well, I don't want to mess our heads with yet more conceptual claptrap, but if you think it's time you really got to grips with Web 2.0, then you're a bit late. But don't worry. Whether it's Web 3.0, the Semantic Web, the Internet of Things, the Real-time Web, or all of these things, you'll know which is which by the end of this book. You'll also have enough feel for these ideas to go do something new and interesting, maybe start the next billion Euro industry.

Most of what I have to write about in this book is the underlying technological patterns emerging right now, such as "Big Data," which is the ability to make value from unthinkably large amounts of data that would have previously languished on arrays of disks and vaults of tape sitting somewhere, potentially gathering dust.

So let's crack on. Let's set the scene for moving beyond the collision of Web with Telco to a place of congruency – the world of connected services.

1.2 Ubiquity: IP Everywhere or Software Everyware?

I spent much of the ink in my previous two books explaining mobile and IP networks from soup to nuts. I'm going to follow the Web hacker's motto:

Don't Repeat Yourself . . . or DRY. . .

Most of the networks stuff I wrote about in my last book remains current, so go take a look. While interesting, it's really not that relevant to this story, so don't worry if you don't know it. However, networking and related protocols still underpin the Web and, for the unfamiliar, I still maintain that a solid understanding of certain principles, like the way HTTP works, will carry you a long way in understanding and accessing new ideas on the Web.

For years the mobile industry got us all in a frenzy about ubiquity. I think slogans were coined about it: "Anytime, anyplace, any . . . something," I struggle to remember. Yeah, we get it. We really do – an IP connection that is! Almost everywhere we go, we can grab an IP connection thanks to the huge and ongoing investment in wireless broadband networks. In most advanced markets, it's difficult to go anywhere without the ability to connect to an IP endpoint: WiFi, 3G, 4G.

What this really boils down to is the ability to make "http://" work everywhere, which is like the "dial tone" for connected services. This is the kind of ubiquity that's become important – software services everywhere – call it everyware![3] Sure, this low-level bit-shifting network stuff still dominates the telco world. And for good reason. That's what they do, and they do it well. But we're not here to consider the "Telco versus Web business model" debate ("we make money and they don't . . . blah, blah.") We're here to explore connected services – making the most of the telco and Web worlds *combined*, or collided. This is not about telcos becoming Web companies, which they will often be the first to say is unlikely. This is about making better, more relevant and future-proof telcos by harnessing and exploiting the technologies, patterns and capabilities of the Web. The opportunities have never been more promising and exciting than today, thanks to a significant evolution of Web technologies and patterns.

What the folk in the telcos often don't get is how the Web world *really* works, especially post Web 2.0. Sure, they get HTTP and all those "Webby" protocols at a distance, but in my experience they have failed to keep up. Where we are today is a world awash with Web-centric software and applications, many of them free of charge, that enable coders to work in even bigger chunks, like writing a novel paragraphs at a time. That might sound crazy, but that's

[3] Thanks to Mike Ellis for that buzzword!

how it works. Moreover, a hacker can take someone else's paragraphs, even whole chapters, and craft them into a new story – that is, an application or service.

And that's where DRY comes into play. If it's already been said, or written, then why say it again? Just run with it. This reflects the idea of re-using stuff as a principle. Much of the progress in Web software has been possible by this re-use principle, which we can widen to include: the open source ethos, things like "social coding", open APIs, mash-ups and so on. Don't worry if these concepts are still alien to you. They won't be for long. Read on.

The essence of ubiquity in the post Web 2.0 era is in the ability to connect via software. It's incredibly low friction. If you can think of an idea for a service, then it isn't long before you can articulate the idea in software and start a conversation with users via their Web browser or mobile app. Talking and thinking in Web software is the new ubiquity. All that low-level stuff that makes it work – the IP stuff – is just taken for granted.

1.3 Six Models for Potential Operator Futures

Having worked on dozens of projects and having had dealings with all kinds of folk, I've concluded that no one really knows how the telco world is going to pan out in the face of current pressures. One thing for sure is that we will always need physical-layer networks. That future is guaranteed for someone. It's not unlike the utilities worlds or the ISP world. Most survive in the realms of efficient operations, value-engineering, rigid cost controls, customer care efficiencies, effective marketing and so on. Some differentiate with niche services (e.g. Heroku), hyper-effective support (e.g. Rackspace), and so on.

But what of these "Connected Services" futures? What might they look like? During a consulting gig for the world's biggest texting infrastructure provider, I postulated some possible future operating models, extending out to the year 2015. This was just to guide their thinking in terms of how their infrastructure products might need to evolve and adapt. Again, I like to follow the lean model of just getting something shipped and then working with real feedback rather than working on grand theories in a vacuum. In that spirit, I threw up these six models as a starting point for the discussion. Once you have something to discuss, ideas can begin to form and crystallize. As part of the backdrop for proposing the models, I identified a number of key trends, challenges and opportunities facing telcos, as shown in Figure 1.2.

These trends led me to propose six models for future telcos (or OpCos as I call them in these diagrams). I don't want to bore you with models for the sake of it. These models will serve to provide us with context as to how and where the Internet and telco worlds might intersect.

These models, shown in Figure 1.3, are not mutually exclusive. Some operators might well end up following all six. There might be more than six. They might also overlap in terms of definition, but each one has a sufficiently distinct set of attributes to make it useful as a potential pattern in the future of telco operating models. I'm not going to play out the models in detail, as this isn't supposed to be a business models book. I'll briefly summarize the essence of each model as a basis for thinking about the nature of connected services for each scenario that a telco might face.

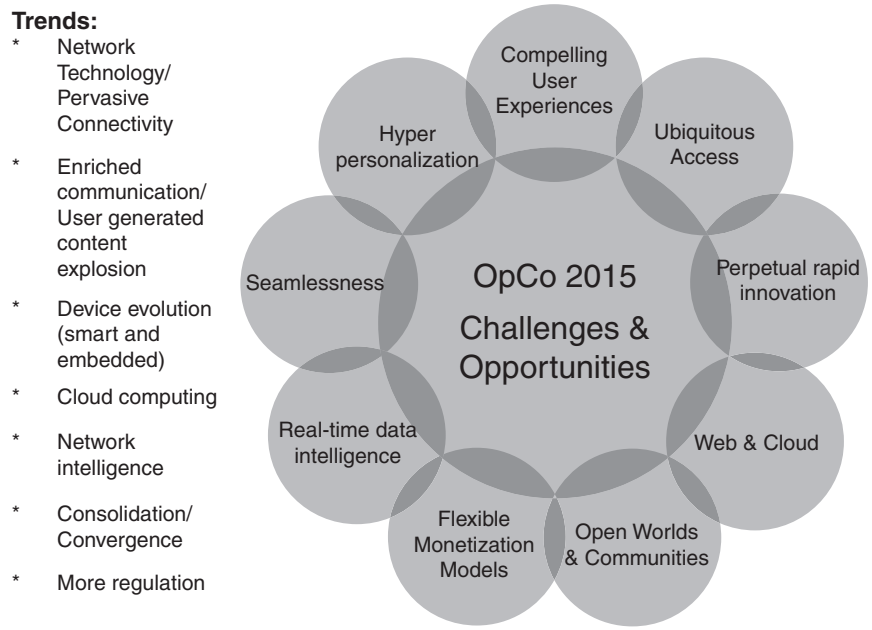

Figure 1.2 OpCo 2015 challenges and opportunities.

1.3.1 Access Provider

This is the pure network play – a bit-pipe. Not to be underestimated in terms of profitable futures, but not the most interesting of models within the connected services context, as it is blind to the services layer running atop. There are some interesting opportunities here for the use of modern Internet software techniques to run a bit-pipe a lot more cost effectively in the OSS/BSS domain, but that's a different story.

1.3.2 Connected Services Platform

The keyword here is "Platform," which is the first time I've introduced this term, but we will be exploring its meaning and implications in depth throughout this book. As you will soon discover, the idea of platforms is absolutely central to modern Web ventures. The idea of this OpCo model is to take the current telco network and enterprise infrastructure and turn it into a platform on which it is easy for other service providers to build new services that can mix and match internal components and capabilities (e.g. Billing) with external components, which means more or less any Web service.

This isn't a new idea, but I mean here something a lot more visionary and aggressive than opening up a few APIs to developers. I mean a radical extension of the business and technology architecture to support Software-as-a-Service and Platform-as-a-Service patterns, which might also include extension into new infrastructure opportunities, like cloud computing. I also

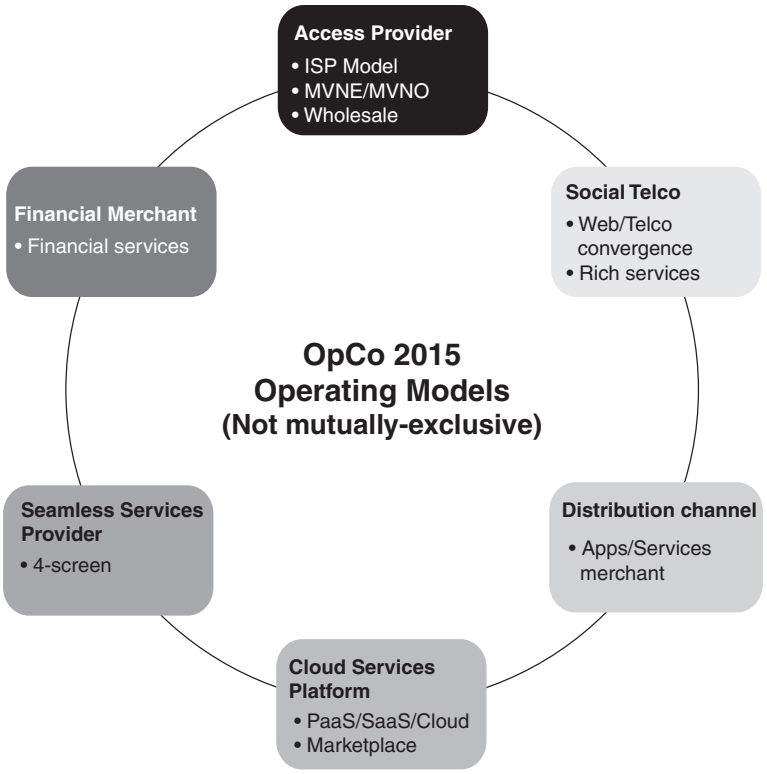

Figure 1.3 OpCo 2015 operating models.

include potential ideas such as Support-as-a-Service. After all, if you're good at supporting customers, then why not at least consider how to turn it into a revenue-generating service as opposed to a cost centre.

1.3.3 Distribution Channel

A major OpCo asset is the extensive user base in consumer and business markets. In theory, a carrier knows a lot about its users, which ought to be a highly exploitable asset in many ways. A carrier also has the means to extract payments and to maintain a relationship with the users. All combined, these are powerful ingredients for a compelling distribution channel, particularly for digital goods, which includes adverts, coupons and even virtual currencies. Many carriers already have retail stores and healthy e-commerce apparatus, all of which are extensible in theory to become low-friction digital distribution channels. The tuning and reconfiguration of the OpCo platform to become an efficient and targeted distribution channel has lots of potential.

A lot of potential for this model is in the adoption and exploitation of so-called two-sided (or N-sided) business models, which means getting money from upstream customers in the way that Google gets most of its money from the advertisers, not the users. Telcos mostly get their money from the downstream users, although this is slowly changing.

1.3.4 Seamless Services Provider

Which carrier hasn't considered moving into adjacencies, like financial services, home services and so on? Many have already tried it, with varying degrees of success. Operators are also moving towards multi-channel offerings: mobile, fixed, broadband, TV etc. Meanwhile, competitors in those adjacent businesses are also moving into their adjacencies, so the nature of competition is shifting all the time.

Inside this milieu of stretching-the-brand offerings, the winners will increasingly be those who can offer the best "joined up" experiences for customers. This doesn't necessarily mean fancy tricks (like music that flows like liquid from the TV to the street to the car). It is more about information and experiences being in the right place and mode as the user moves from one service to another. It's about giving a compelling user experience across a wide portfolio of services, which is not easy. As we shall explore later, the idea of the "Right-time Web," is all about the right information at the right time and place.

We shall get to this in due course, but "joined-up-ness" will increasingly only be possible via various syndicated connections on the Web. Telcos do have a lot of information about their users, but not as much as they think they do in comparison to the Web, where users increasingly leave substantial trails of digital footprints. In other words, a large part of the user context that a seamless provider will need access to is situated and evolving on the Web. To illustrate this point, I include a diagram (see Figure 1.4) from my previous book, just to emphasize the shift in "centre of gravity" from the telco to the Web.

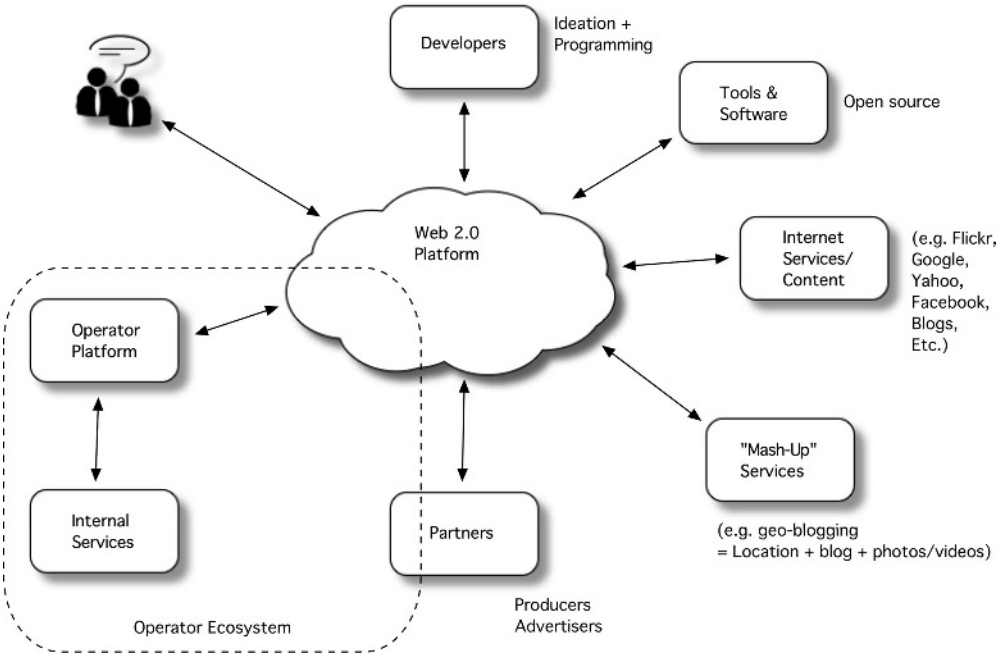

Figure 1.4 Shift of gravity towards Web platform.

Sadly, telcos spent years talking about "Context-aware Services," only to lose the march completely to the Web. Even now, many strategists inside operators still don't get that "Context," is what Web brands are all about, ever since they discovered the ad-funded model. Relevancy is king in that world. Relevancy is all about context. On the other hand, most telco services operate in exactly the same way no matter the user context. Making a call is making a call is making a call – it seldom adapts dynamically to the user's context. Going further still, the networks are oblivious to the content of those calls, which is like Google not bothering to exploit the content of search queries – can you imagine?

1.3.5 Financial Merchant

What does a telco do? One thing it does well is to add up lots of activities (rating) and then charge the user (charging), telling them what to pay (billing). For ages this capability has only been thought about as an adjunct to telephony, not a capability in its own right, capable of being extended to all kinds of ventures. Sure, various visionaries have seen the potential of extending the "money machine" to other applications, but they are generally hamstrung by the inflexible nature of the IT infrastructure used to build these systems.

This is in total contrast to the likes of Amazon and Paypal whose payment systems were born in the world of Web software ubiquity – just another application running on the LAMP stack, or similar.

However, with a similar approach, operators could substantially reduce current costs of running the money machine whilst freeing up considerable flexibility to pursue new models in an agile fashion. A lot of operators are keen to pursue this model, notwithstanding various regulatory restrictions and value-chain complexities that might impede their ambitions.

1.3.6 Social Telco

This model says that an operator will not sit back and concede its previously central role in communications to Web players. I think that the old-school thinking of "build a better internet" is over, whether it's called IMS or Wholesale Application Community (WAC).

Mobile phones have always shipped with address books. Unspotted by most, this combination of phone with address book made a unique communications device at the time (see Figure 1.5).

That's right, our mobile phones were actually *social networking* devices because they contained a list of our friends and associates along with their contact details, allowing social connections (calls and texts) to form, which is the essence of a social network. Conceptually at least, there is no difference between this address book of friends and the list of friends on Facebook, or LinkedIn. Mobile networks plus devices were the original social networks, but woefully under-exploited as such. Sadly, the address book didn't become the nexus of connectivity that it might (and should) have done, even though some of us (like myself) were talking about and demonstrating this idea from very early on. I wrote extensively about this in my first book, describing how the address book could and should become a fulcrum for all kinds of useful services. I even suggested the appropriate Web technologies and protocols[4] for making this happen.

[4] Such as the emergent Friends of a Friend (FOAF) data format.

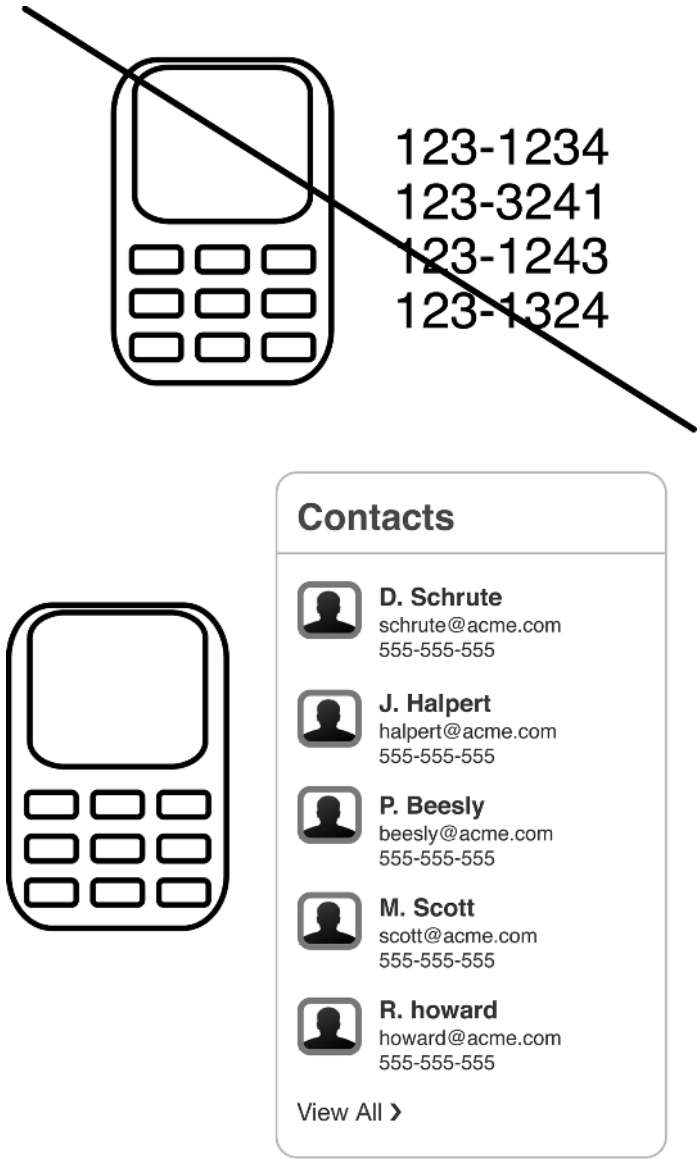

Figure 1.5 Phone as a social networking device.

Operators sat back and did nothing, mostly. Again, this is said non-judgementally. This book is about the future, not lamenting about past and gross oversights and failures (e.g. WAP, MIDP) and missed opportunities (e.g. Social Networks) of the telco world. Pity! What a missed opportunity that was. Or was it?

This operating model says that there's no way that the socially-relevant services bet is off for telcos. I think that tier 1 players will have to embrace a lot of the current Web trends in order

to remain competitive and relevant. It's not like they're sitting back anyway. Millions gets spent on various R&D programs and acquisitions every year, although often poorly executed. Some telco players and MVNOs will push this envelope hard and might set the standard. Who knows, we might even see a "Facebook Mobile Network," or equivalent, which exists entirely within the Facebook ecosystem. It is certainly possible and I have proposed such ideas in my consulting gigs.

1.3.7 Start Thinking Platforms

Whether it's distributing digital goods, enabling financial transactions or exploiting social connections, hopefully you might have noticed that in all these cases the network becomes a little bit more intelligent than just a switch, connecting one device to another. There's all kinds of intelligence in the connection, leading some commentators to posit the meta-model of "Smart Pipes," which I find somewhat oxymoronic as a term. I prefer to think of platforms, which is an exciting theme on the Web that I'm going to explore from multiple angles throughout this book. However, before we do that, let's just think a little about what our "telco platform" might look like for any of these OpCo models. Figure 1.6 begins to tell the story of operating a platform rather than a network.

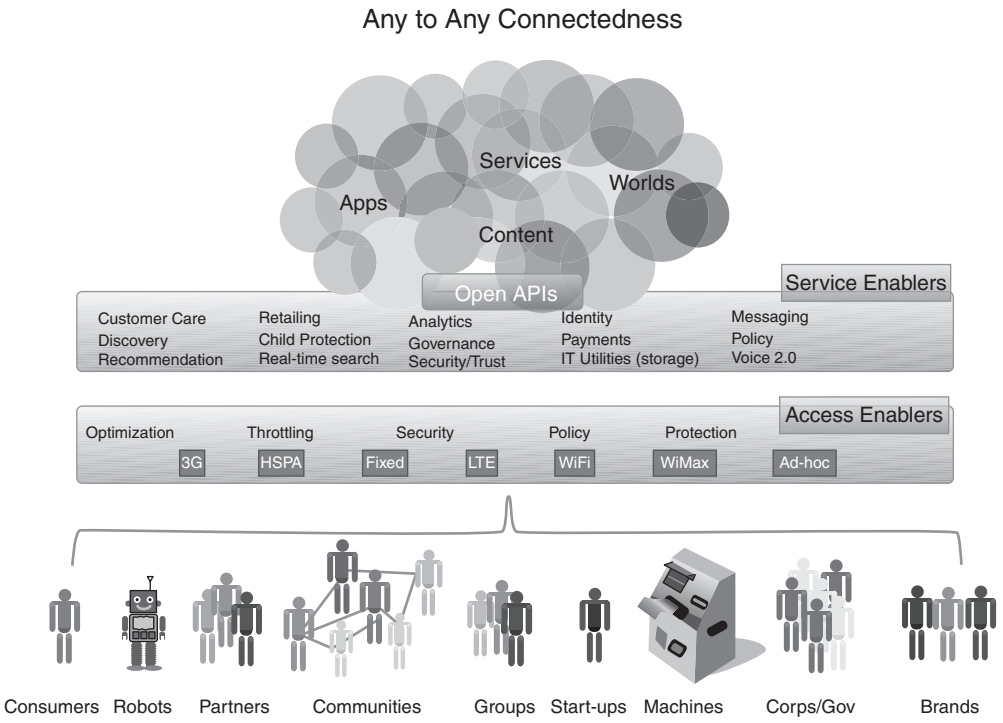

Figure 1.6 Any-to-any OpCo services platform.

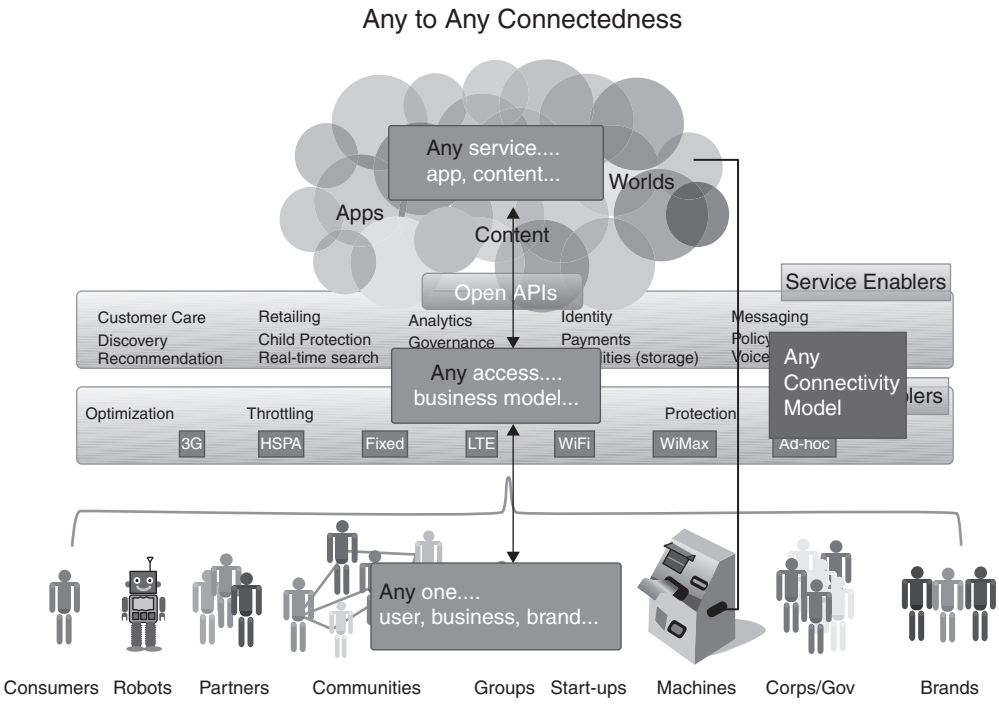

Figure 1.7 Any-to-any business models.

What the diagram shows is four layers:

1. **The users** of the services either running on or accessible via the platform, such as consumers, brands, communities and even machines.[5]
2. **The access enablers** that telcos traditionally think of as the network (e.g. HSPS, LTE) including support services like compression and security.
3. **The service enablers** that really add the intelligence to our services, enabling routing, personalization and all other manner of value-added functions in the service chain.
4. **The digital content** which includes applications and services that users want to access, either running on the platform or via the platform.

This platform model is how telcos should be thinking, no matter what the OpCo future they envisage. Platforms automatically bring to mind accessibility – allowing other parts of the value chain to seamlessly integrate with the telco's capabilities. This leads to a generic business model architecture, as shown in Figure 1.7.

Ultimately, the telco should have an ultra flexible and low-friction platform that enables agile construction of high reliability services that connect anyone (user, business or brand) with any digital service (app, content, etc.) via any access technology and within any business

[5] Note that "Robots" in the diagram means software acting on behalf of people, not physical robots.

model. What you might notice about the platform diagram above is how it still appears to be very telco centric, despite my earlier point about the shift of gravity towards the Web. Of course, as a telco, you are only concerned with your platform and your business, which is largely what the platform diagram depicts.

However, it is in the nature of the connectivity to the user layer and the services/content layer that the smart telco will build interfaces that work seamlessly to exploit the power of Web platforms. For example, if it makes sense to provide a service that enables brands to connect with mobile Facebook users in a way that only an operator could exploit, then this any-any-any platform model should be constructed to support such a possibility. The key point is that the telco should focus all of its efforts on building a platform, not a collection of stove-piped services that lack the flexibility to be deployed in other ways. Having a powerful platform allows the telco to engage in new types of business models with a wider set of partners in a wider number of value chains and networks.

1.3.8 Execution

Don't get too caught up in the opco models we've just looked at. They're not cast in stone. However, they are certainly plausible and a useful starting point for our exploration of the execution question. They do overlap, although each one has a fairly differentiated strategic emphasis. So, how does an operator evolve to any or all of these models? How do they execute?

Not by following business as usual, that's for sure. Here are some default behaviours that certainly won't do in the connected services universe that we now find ourselves in:

- Relying on current and traditional telco suppliers to follow the curve of innovation and adoption = fail.
- Relying on traditional customer insight models to guide service design and deployment = fail.
- Relying on the same key people who built the business for the past decade = fail.

To shoot for success, or even gain a chance of starting in the right direction with any of these models, with the possible exception of the first,[6] it is imperative to gain an insight into the new technological and socio-technological patterns emerging on the Web.

Make no mistake that apart from the first opco model (Access Provider), all of the other models are executable by non-traditional competitors, such as born-on-the-Web ventures, to varying degrees. Moreover, they are busy pursuing it, right now as you read this book. And they are mostly operating in the zones of innovation freedom that telcos find very hard to enjoy, not stuck in the limbo zone.

1.4 "Follow Me" Web – Social Networks and Social Software

Social networks are huge. They are getting bigger and better. For many Web surfers, social networks are the focal point of their digital lives. Facebook has over 500 million users, getting

[6] This model almost says to carry on business as usual. However, don't overlook that potential to use Web methodologies, technologies and patterns to enable operational innovation with this model.

bigger every day. Twitter has over 106 million users, increasing 300K per day at the time of writing this book. These figures might well change, but both of these "start-ups" are aiming for 1 billion users. That's huge, if they can get there. Some commentators have pointed out that the population of Facebook is bigger than the population of many countries combined. Facebook even has its own currency!

Facebook and the various social networks out there are more than just websites or "communities" (another popular "C" word). As Facebook founder Mark Zuckerberg said:

> "You don't make communities – you enable them."

The word "enable" is the key to Facebook's technological approach and to their success. Operators might do well to emulate it. Bit by bit, Facebook has provided the tools for users, partners and developers to enable various community behaviours that flourish on the Facebook platform. Facebook has exploited the network effect at every level, as has Google, Yahoo!, eBay, Skype, Wikipedia, Craigslist, Flickr and many others. MySpace didn't, failing to provide the access hooks that Facebook did, which is why it was rapidly over-shadowed by Facebook as Facebook's network effect kicked in. To quote my respected associate Amy Shuen in her insightful book: *Web 2.0: A Strategy Guide*:

> These enterprises have strategically combined different kinds of network effects - including direct, indirect, cross-network, and demand side, to multiply the overall positive impact of network value creation.

I wrote extensively about Network Effect in my first book and will return to it frequently in this book, but here I want to emphasize the underlying infrastructure for network effect in a social network, which is the Social Graph.

The social graph is a mathematical map of who's connected to who, see Figure 1.8.

The concept is easy to appreciate, but here's the really important idea about the social graph . . .

Increasingly, *the social graph is the Web.*

What do I mean by this? When the Web first materialized, and for much of its early growth, the gold rush was about connecting people with pages of text. Early Web protocols, like HTTP, were good for this purpose. They still are. These early pages were static, which means authored and created in advance by page creators, much the same way that every word in this book is committed to paper before you read it. It doesn't adapt or change according to the reader's needs and feedback.

Later on, programmers realized that HTML pages could be generated on-the-fly, or dynamically, by programs. That's what happens today with most websites, and various website software frameworks have been invented and gradually improved to make this easier. The page doesn't exist until your browser asks for it. A program on the server then generates the page and dishes it out to your browser. These pages can be personalized, which is how you can now use the web for personalized services like banking, shopping, travel bookings and so on.

In this transition from static to dynamic content and from publishing to interactive services, Web 2.0 came into being. A key theme (or meme) was the growing importance – and voice – of the users. No longer just a one-way publishing experience, the move to dynamic web allowed users to transition from consumers to producers of content. They could add their content, as

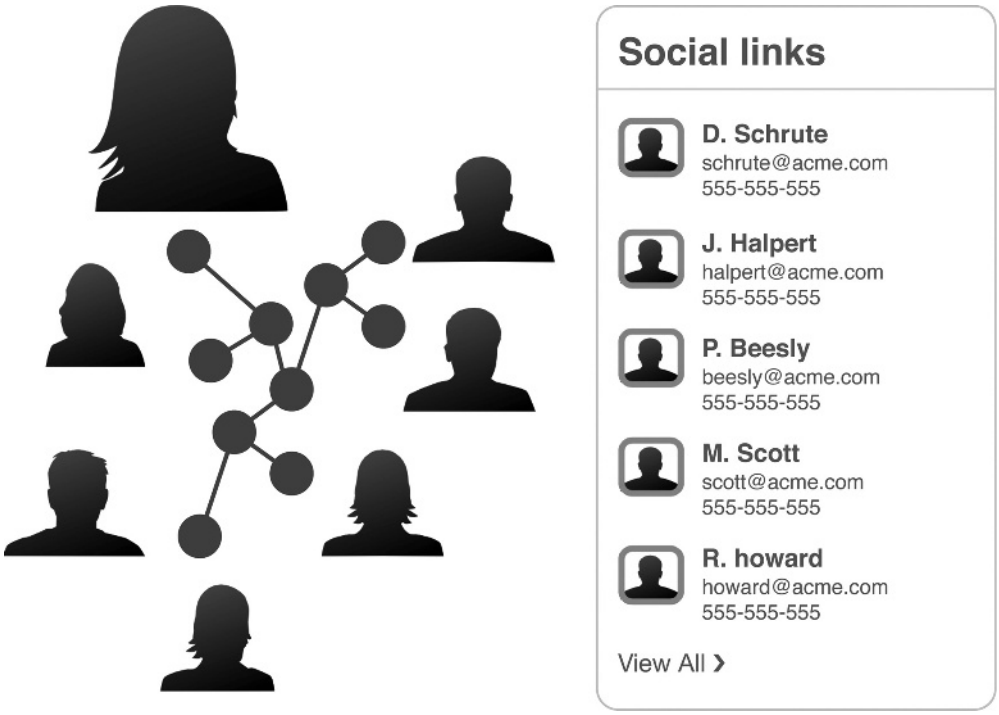

Figure 1.8 Social graph – who's connected to who.

sparked largely by the blogging revolution at the beginning of the 00s decade, followed by social networks, like MySpace.

Once users entered into the community of contributors, they rapidly took centre stage. And this is when the Web got social. It became rapidly clear – and is still rapidly evolving – that the connections between people really matter in Web services. People, and the attention they pay to each other in various modes, drive the network effect, which is a vital precursor to generating value on the Web.

Facebook and social networks have taken this idea to the next level. They have created the platform and tools to enable digital social connectivity, not only in ways that Facebook dictates, but in all kinds of new, unexpected and unintended ways. In other words, Web platforms, like Facebook, are more than just platforms for connectivity and services – they are also *platforms for innovation*.

One of the ways that this works is via the indirect network effect, which is why Facebook allows other services to exist on its platform, allowing these services to exploit the social graph. It is a brilliant strategy because it solves one of the key problems that all digital services suffer: discoverability.

The social graph is multi-dimensional. It doesn't just graph who knows who, it also graphs similar interests, habits, likes, dislikes, event attendance, fandom and so on. This graph of discoverability is essential to the success of a number of the operator models that I laid out in

the previous section (see Section 1.3 Six Models for Potential Operator Futures): Distribution Channel, Connected Services Platform and Social Telco.

And it doesn't stop there. Facebook has recently moved its attention to exploiting the cross-network effect by opening up its Social Graph API, which is a protocol for external services to mesh their social graphs with Facebook's social graph (on a per user basis). Just as the cross-network connection for SMS sparked the texting boom in the UK (and eventually elsewhere), we can expect similar dynamics and value-creation from Facebook's move and similar initiatives like OpenSocial.org. This move to open and shared connections is accelerating. Facebook Connect is a service that allows a user to log-in to another service (which might have nothing to do with Facebook) using his or her Facebook log-in credentials. It turns out that this usually results in a much higher sign-up rate to the new service. In some cases, new sites are not even bothering with their own login system – they simply piggyback on Facebook, Twitter and other open authentication systems.

Similarly, Google had already released its "Friends Connect" service, which enables any web service to embed social graph information built on Google's social services. These are all examples of a kind of inverse-platform effect whereby one platform (such as Facebook) extends out to another (which could be an operator's portal) to become an on-ramp or enabler for that platform, which, once adopted and favoured by users, is hard to undo. This is a kind of "Trojan Horse" strategy that some Web players, like Google, have been keen to push hard for some years now.

The social graph is also dominant in Twitter, another service that simply has no value without the concept of social connections. It is important to grasp that the formation of social connections in Twitter and similar services is both explicit and central to the function of the service. A Twitter user has to explicitly – and publicly – "follow" another user. A user declares to the Twitter platform, and all of its users, that "I follow X," and "I follow Y." Once these "follow me" connections are formed, the Twitter platform keeps track of what each user does and then makes sure that this activity is passed along the appropriate parts of the social graph towards the user's followers.

This is unlike telephony where a user can connect to a number ephemerally and in private. There is no lasting connection and no concept of "following." The social graph in an operator network is entirely incidental to its operation. In social networks and increasingly the (social) Web generally, the social graph is pivotal to the service. Social telcos and Connected Services Platforms will have to find a way to put the social graph at the centre of their businesses. This might be a telco-centric social graph, a social network's social graph, or a hybrid of the two, which is the more likely scenario. That said, operators must push hard to innovate on top of the social graph that is inherent in mobile network operation.

As it turns out, building large scale software systems around social-graphs is not without substantial technical challenges. Each time a user carries out an action on the platform, the platform has to propagate this action along the social graph in its various dimensions, and in a timely manner. In the case of Twitter, this happens in near real-time. Moreover, this challenge rapidly mushrooms as users along the graph forward actions and outcomes to their followers, creating a cascade of updates. This requires constant evaluation of connections in the graph, which turns out to be a thorny computer processing problem capable of bringing Web servers and databases to their knees very easily, especially as the network grows with exponential impact on resource requirements.

Twitter was inspired – perhaps forced by performance concerns – to create its own database technology, called FlockDB, just to solve this problem at scale. I mention this because it reveals another interesting and dominant technological pattern in leading Web ventures, which is the drive to create novel types of technologies to maintain a competitive advantage. Operators are currently not equipped to handle such challenges. Despite operating very technically advanced platforms – that is, cellular networks – telcos are not technology companies. They lack the sorts of people who can build new technology platforms. For example, it is virtually unheard of for a telco to build a new database technology, whereas it has happened often in recent years on the Web, with Google, Amazon and their cohorts cooking up all kinds of new storage techniques and solutions (see Section 4.2 Some Key Examples of Big Data).

That said, operators will argue, quite correctly, that they haven't had to invent new types of technology because their suppliers have been doing this quite successfully for decades. Operators will also point out, quite correctly, that they have been addressing large-scale processing and storage problems for decades too, perhaps the fathers of "Big Data." All this is true. However, it misses an important point, which is that without the in-house expertise to recognize technological opportunities and without the pressure of finding new solutions to new business problems – that is, the pressure of product innovation – operators wouldn't know what opportunities they have missed as a result of not being technology companies. For example, it is clear to me that a different type of operator staffing and culture would have spotted the social networking opportunity some years ago and we would, contrary to my opening section (see The C Word, Again) be talking about telco giants and Web giants in the same breath. Alien visitors would indeed see telco logos dominating the Web.

To give another example of an opportunity that is difficult to exploit without at least some technological insights, one of the biggest recent advances in Web technologies has been the advent of "Big Data," which is the use of massively scalable data storage and processing engines to find patterns in data that would have previously been hard to find, economically and technically. It turns out that the ability to process unthinkably large amounts of data is perhaps the key to unlocking new value streams for Web ventures. The same probably applies to Telcos, which is a theme I shall explore when we consider Big Data patterns in depth in Chapter 4.

1.5 What are Platforms and Why are They Important?

I have already used the word "platform" quite a bit in this chapter, but without really exploring the idea in any depth. Let's do that now, because, as I hope you're beginning to appreciate, "platform thinking" has a place in telco futures. Platforms aren't a difficult idea to describe or understand, but can be hard to execute, often deceptively so. The software platform war probably began with the desktop operating system (with early winner Microsoft) and now continues with social networks like Facebook and with other platforms like the device platforms of Android and iOS.

A platform is different from a service or an application, in that a platform is something that other people build services and applications on top of. Platforms are not new. The obvious and pervasive example is an operating system, like Windows. However, there are other examples, perhaps a bit less obvious at first, like the Visa network and the Playstation 3. These are examples that are often quoted in the business literature, such as the intriguing writings of Andrei Hagiu, which I urge you to read.

In his fascinating book "*Invisible Engines*," which I highly recommend, he (and his co-authors) argue that in order to understand the successes of software platforms, we must first understand their role as a technological meeting ground where application developers and end users meet.

To me, this is the crux of the matter when we talk specifically about platforms. For now, I am not really concerned with examples like Visa network, which is an N-sided platform (users, retails, banks). Of course, such platforms might well prove to be of interest to "Financial Merchant" operators (see Section 1.3 Six Models for Potential Operator Futures), such as when exploring mobile wallet ideas and other financial-intermediary solutions. However, in this book, in order to narrow scope, I am going to focus mostly on software platforms that involve the Web, with software developers as one of the key platform users. Later on, we will get into the detail of who developers really are, in order to dispel the mythical notion of developers as folk who sit in messy bedrooms (or garages), more interested in breaking systems than creating value through enterprise where they exploit their works of creativity and utility. Developers are, and always have been, highly entrepreneurial and, increasingly, creators of massive "value nets" around their wares. They are rapidly becoming the industrial moguls of our age.

What initially deters operator stakeholders from exploring platforms seriously is when the ugly word "free" crops up, as it often does early in the conversation. Apple, Microsoft, and Google, for example, charge developers little or nothing for using their platforms and make most of their money from end users; Sony PlayStation and other game consoles, by contrast, subsidize users and make more money from developers, who pay royalties for access to the code they need to write games. More applications attract more users, and more users attract more applications. And more applications and more users lead to more profits.

Importantly, at least for this book, I'm only going to explore platforms that have the potential, at the very least, for an associated business model to emerge – a means to convert platform usage, however that gets metered, into revenue and profits. Some kind of commercial (and contractual) relationship is necessary and inherent in using the platform, both as an end user and a platform application developer. So, this rules out the open Web itself and similar infrastructure platforms that are, like most highways, completely open and essentially free of commercial restrictions. (This isn't entirely true, but we don't need to debate the point here.)

If we consider Facebook, for example, then a developer wishing to develop a Facebook application is required to sign up to Facebook's developer terms and conditions. These controls are obviously important for Facebook to manipulate the use of the platform to its commercial advantage. That said, successful platform ecosystems are always symbiotic. Both the platform provider and the platform developer stand to win if both are successful – a classic win-win dynamic that hopes to exploit the network effect of attracting more users to the mutually beneficial alliance of platform owner with platform service developer. In the case of Facebook, some of the applications, like the Zynga network of social games, have proven to be more profitable than Facebook itself.

Experience shows that an important realization about platforms is that you've got to know that you're in the platform business in the first place and then be really committed to making the platform work, which is usually synonymous with lowering friction of usage on both sides, especially the supply side. That sounds obvious, but not realizing that you're in the platform business, with all that it entails, is a common mistake. It's important to understand that for a platform to be successful its owners have to cater very well for the developers who want to

build stuff atop of the platform. The history of software (and computers generally) is littered with the corpses of venturers who thought that they were in the product business when what they actually had was a platform. A recent example was the innovative product called Groove, which was an enterprise collaboration product that didn't require a central server. It used so-called peer-to-peer (P2P) technology, first brought to our attention by illegal music sharing sites like Napster.[7] Instead of music, Groove used P2P to keep information and content in sync directly between software instances running on separate desktop machines. For example, a team of project workers were able to share files between their PCs without a file server.

The team at Groove focused all their efforts into the central product experience. However, lots of other (external) developers were attracted to the product as a platform. This was because the problem of keeping data in sync between machines turns out to be a highly useful, yet thorny, problem to solve, and one that most developers would rather avoid tackling.[8] There were a queue of developers wanting and waiting to build their wares on top of the "Groove stack"[9] in order to achieve two things:

1. Offer their own killer product without the headache of having to write and maintain the awkward P2P synchronization infrastructure.
2. Benefit from the network effect of seeing (hopefully) more users on the Groove platform as a result of adding extra features and services that strengthened the core offering without involving the core team. Of course, in return, Groove could benefit from the indirect network effect of these other users becoming more interested in and committed to the Groove product as a result of these companion services.

When Groove began to understand how developers were viewing their product as a platform, they failed to take advantage of the opportunity. They remained wedded to the idea that they were really in the product business. They viewed the third-party stuff as a kind of "sideshow", rather than something that could have been core to their business strategy. In the end, the core product failed to gain enough traction (because it was missing the features that the third parties might have brought to the users) and the opportunity was missed. The third parties were denied the chance to ride on the platform because of prohibitive pricing and terms. The business model was a product model, not a platform model, even though it eventually became obvious that Groove should have been a platform. By then, it was too late.

There are numerous examples of half-hearted attempts to be a platform player, which almost always exhibit the same tendency, which is to alienate the very developers who could make the platform successful. Operators have excelled at alienating developers with unrealistic commercial terms, high-friction interfaces and various forms of condescension. As I have pointed out to many senior telco stake-holders, were they to treat customers the same way they treated developers (who are a customer for the platform), they would be out of business and probably get fired. For example, imagine charging customers to enter a shop, or asking them

[7] Before it went legal of course.

[8] Have designed sync products myself. I can tell you that I'd rather stick pins in my eyes than ever try it again.

[9] When I talk of stack, I mean the actual software that underpins the Groove application. The "platform" of Groove is not just the stack, but the stack combined with the users and a means to add additional services on top of the stack, making them available to all the users.

to wear a tie before entry. Imagine ignoring them when they got in and then asking them to take one of those tickets you see at the deli-counter lines. But this is how developers have been treated by operators for years. It's no wonder that many of them hate operators with a passion and can even get religious about supporting any technology that makes telcos irrelevant. I should add that whenever I've worked with or for operators, it has only ever been in the role of "developer," trying to fix the problem from the other side – trying to convert operators into applications and developer-friendly companies. As one of the pioneers in creating mobile applications in the 1990s, I suffered too many headaches bashing my head against the wall of operator ineptitude towards developers.

With platforms built on the Web, there is often a tension between a services play and a platform play, similar to what I described with Groove. Sometimes the venture can't decide if it's really an application or a platform, resulting in poor attention to the developers, the **only** folk who might be able to make it work as a platform. Many of these ventures suffer the same fate, which is aggressive disruption from a copycat who gets the platform play right. Joel Spolsky documents a number of such examples in his article about platforms from which I quote:

> The best way to kill a platform is to make it hard for developers to build on it. Most of the time, this happens because platform companies either don't know that they have a platform . . . or they get greedy (they want all the revenue for themselves.)

One reason for the tension between application and platform is the need for the venture to promote its own platform with an application that will attract enough users to gain critical mass in the first place. After all, without a user base, developers might not be interested in developing for the platform. This is not always true, as I said during my presentation of this topic to Mobile World Congress in 2010. I pointed out that a platform very often needs "cool power" in order to attract developers, which itself promotes the platform to end users and to other developers keen to join the action and fun. Twitter and iPhone are two examples where the shear coolness of the platform had developers eager to build apps.

When I'm asked what does cool mean, another of those thorny "C words," I usually say, not entirely in jest, "if you have to ask, then you won't get it." My answer seeks to expose the fact that software development is a very broad spectrum of activity and people, and not everyone is going to "get" cool. Anyone can open a computer and start coding. It's mostly free and doesn't really require permission. This is just like playing a musical instrument and we can draw an interesting analogy here. Music can be both a playful, artistic, creative endeavour and an industry making billions. Composer John Williams makes vast amounts of money out of writing scores for films. However, much of the innovation in music, like the invention of Blues guitar, came from individuals who had the tools and inclination to experiment. Similarly, coding software is just as much a creative pursuit as it is a commercial endeavour. At the creative end of the spectrum, new and shiny tools will attract a lot of attention and generate a lot of innovation. This innovation is then funnelled by various entrepreneurial processes into money-making industries. So, very often we can substitute "cool" with "new." In the case of a telco, we might suggest to give developers something they just haven't seen before or can't be done anywhere else. Sending a text message hardly qualifies as new and exciting. However,

location-finding, had it been exposed to developers way back in the early 00s, would have been a very interesting platform enabler to offer developers.[10]

As I have shown earlier in the opco models for the future, becoming a platform player is going to be key for some operators. The challenge here is understanding exactly what makes a successful platform. Operators tend to focus, quite proudly, on their assets: "we have this, that and the other." Frankly, nobody cares unless the assets are easily usable, like Lego bricks. If ever there was an example of high-friction, it's probably working with an operator. Even attempts to lower the friction have mostly failed. This is usually because operators try to run their platform efforts by following a regular telco playbook. It's not enough for the usual suspects (e.g. Marketing) to run a platform business just like any other. The people running it have to understand first-hand what running a platform is all about. Developers are generally very smart people with certain expectations from technology that operators often fail to grasp. It's a bit like trying to run a wine-tasting event for expert wine tasters without knowing anything about wine. Who would dream of it? Yet this happens frequently with operators and their developer efforts. This has the unfortunate effect of causing failure that is interpreted by senior stakeholders as being a sign that platforms don't work.

Most successful software platforms have exploited direct and indirect network effects between applications and users – more applications attract more users, and more users attract more applications. We are beginning to see this with the iPhone and iPad, probably more so with the iPad which is marketed and purchased for its diverse number of applications, which add up to significant perceived value in the eyes of potential consumers. Nurturing both sides of the market helped Microsoft attract thousands of applications and hundreds of millions of users for its Windows platform. During early telco developer programs, I pointed out the stark contrast between how Microsoft courted developers and how operators did it, even taking into account their inexperience with developers. Frankly speaking, the contrasts were embarrassing. Sure, Microsoft has years and years of start on the telcos, but that isn't a developer's problem. If I were to launch a new car into the market, I can't ask consumers to overlook basic shortfalls, like missing window-wipers, because Ford has a head start on me. Again, such excuses are part of a culture that really seems to say that operators don't really want to be in the platforms business. I believe that the luxury of such a stance is rapidly disappearing.

The same business strategy of exploiting indirect network effect worked for Sony PlayStation in games and Palm in personal digital devices (before they lost their way[11]). But some software platform vendors have done little to help application developers make a success of the platform. The canonical example is IBM, who neglected developers with their mainframe operating system. The same is true today for many ventures whose businesses are built on software platforms in the technical sense, but for dedicated devices such as ATM machines – and operator networks – which remain closed and underexploited as commercial platforms. Indeed, as we all know, operators did their hardest to keep their IT stacks from becoming platforms, keeping them firmly closed behind "walled gardens."

I will explore various platform opportunities for operators in some depth throughout the book (e.g. see Section 9.1 Opportunity? Network as a Service). Meanwhile, in this section I

[10] Location finding has been possible on networks for a long time but was never exposed because operators couldn't find a business case, or, as was the pursuit back then, a "killer" application.

[11] Platform businesses are just as susceptible as any other business to the rigours of the open market and the usual challenges of sustaining a business lead.

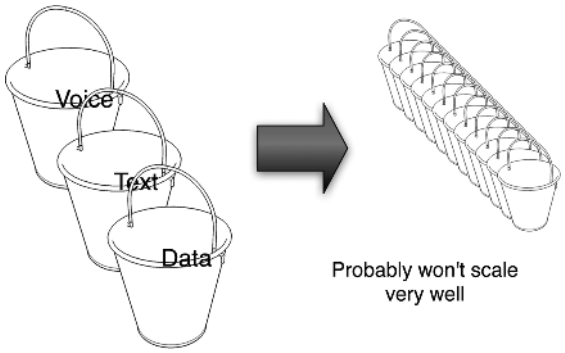

Figure 1.9 Three big buckets replaced by lots of smaller buckets.

will briefly explain the basic configurations for platforms, so that the key concepts are clear before you read the rest of this book. Let's begin by thinking about some of the rationale for platforms. Apart from extension through acquisition and merger, which is an entirely separate type of innovation beyond this book, operators will be forced to find new sources of revenue. The current sources are the big three: voice, messaging and now data. These are three very large revenue buckets that are heavily seeded by the licensed oligopolistic nature of the telco industry. As the license advantage wanes due to technological and regulatory disruption of one form or another, operators, just like every other business, are being forced to find new revenues through innovation. However, the new sources of revenue, at least until "the next big thing" is discovered, are likely to be a range of service innovations that generate revenue in smaller buckets than the big three sources, as shown in Figure 1.9.

The challenge here is that operators are heavily structured and aligned to support the big three buckets. Creating new services quickly enough is a massive challenge, especially if each service is treated with the same mindset as the big three, requiring similar business processes that are hardly built for the speed and flexibility that new services often require. Ask any senior stake-holder how he or she is getting on with new business revenues and they will almost certainly bemoan high revenue aspirations with poor delivery thus far. Little wonder really. Without a "new service factory" pattern, how can an operator be expected to develop substantially new services at all, never mind ones that are profitable? An alternative view of the problem might be shown by Figure 1.10.

This is a new approach. It says that rather than focus on building new businesses per se, focus on building new capabilities that can be turned into businesses both by the operator directly and via a platform play wherever possible, especially if both can be done with similar efficiencies and processes, almost like a "two-for-one" deal.

1.5.1 Platform Patterns for Telcos

Let's boil the background business problem into its component parts in order to understand the potential advantages of platforms before reviewing the different patterns to address the problem.

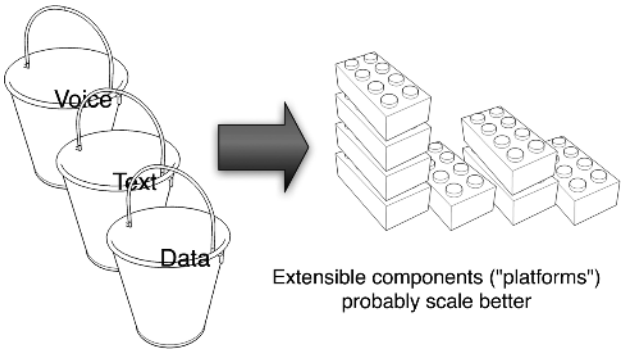

Figure 1.10 Three big buckets becomes lots of building blocks.

- **Problem 1:** How to try out, build and deploy "connected services" utilizing the low-cost and high-speed efficiencies of Web.
- **Problem 2:** How to create services that easily combine Web 2.0 services with telco services.
- **Problem 3:** How to continually evolve a live service to keep fresh and relevant.
- **Solution:** A connected services platform play.

What does a connected services platform look like then? There are a number of key architectural patterns for Web platforms, so let's summarize them here. We will get into the details throughout the book. The first principle of platforms is to evolve from a closed technology stack used to provide proprietary services towards a more open stack that can be used by other parties, namely developers.

1.5.2 Marketplace and Service Platforms

On the left-hand side of Figure 1.11, I'm showing the typical operator pattern used to deploy services today. It is tried and tested and can't be argued with from a historical perspective as it has served its masters well, generating billions in revenues around the core big-three services: voice, messaging and data. At the bottom of the diagram we have the business, which is all the stakeholders and their cohorts who go about the enterprise of bringing ideas to market. They do so via projects that get instantiated on the existing IT and Network infrastructure, which I'm calling the Technology Stack. Typically, any project requires an iteration to the stack. A new service can only run on the stack with a degree of iteration to the stack's IT components, ranging from configuration changes to wholesale sourcing and installation of new solutions (invariably called platforms[12] from an internal IT perspective, e.g. "Voicemail platform"). These iterations take time and money, from little to a lot. The exact amounts aren't

[12] I thought that I should mention this because the word platform is obviously widely used within a variety of contexts. Some telco IT folk reading this book might think that they're already delivering platforms. Please read on to understand the important distinctions.

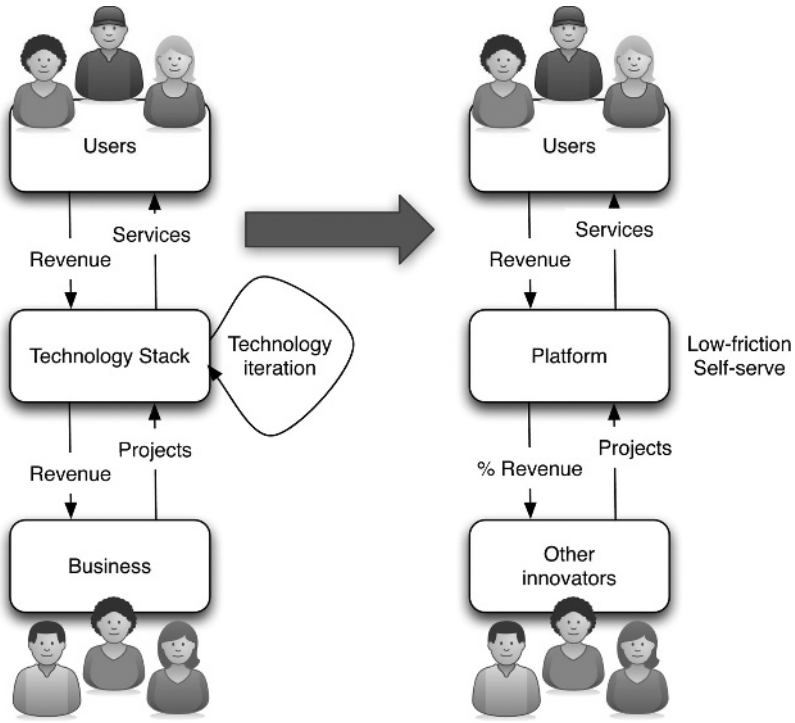

Figure 1.11 From closed technology stack to open platform.

of interest, just the principle of iteration to the stack, which is not without a lot of friction of all kinds (organizational, political, financial, methodological and so on).

The IT division will spend a good deal of its resources in just managing the iteration process. However, it is not without rewards of course. Eventually, when the first iteration is complete, a new or modified service is ready to run on the stack and can be delivered as a service to the users. Of course, all kinds of other business wraps might be required, such as marketing and retailing, but these too will have impact on the stack, requiring their own feature outputs from the iteration process. If the service is successful, the users will pay for it and pass revenues back to the platform to be enjoyed by the business, including new iterations and profit margin.

Moving over to the configuration on the right hand side of Figure 1.11, what if we can alter the approach a little. Take the IT stack and modify it in a way that some of the key enabling components, in whatever configuration makes sense, expose themselves to the outside world via a method that is easily consumed by external software systems. A standard pattern here is the use of Application Programming Interface (API) approach that was first seen inside of operating systems in order to achieve the same goal, which was to empower external developers to build new services on top of the core operating system.

The key is to allow external innovators to use the platform with as little friction as possible for both parties. The platform owner should not have to iterate the internal IT stack to allow an external service to access the platform APIs. The developer should not have to request or wait for any changes to the stack in order to use it as a platform. As much as possible, the goal

is to achieve self-service. Apart from entering some legal agreement between the two parties, also in standard format, the innovators should be free to consume the platform services within the bounds of the API specification (including terms and conditions). The mantra of platforms is "low-friction." This is a theme that I will emphasize often.

With this platform configuration, external innovators are free to run their own projects, pursuing their own ideas, which may or may not be congruent with the platform owner's business ideas. They write software that consumes platform services via open APIs and deliver services using a combination of their software and the platform-capabilities. In many cases, the platform itself is also the vehicle of delivery. In other words, users of the service are also users of the platform. In the case of a telco, this would mean that only Telco-X users can use the services built by external companies via the Telco-X APIs. However, we shall see that this isn't the only configuration. Users could well be indirect users of the platform, in the way that Google Maps are often consumed by a large number of users via channels outside of Google's mapping website. This "Mash-up" configuration isn't unique (it was around before the Web) but is increasingly prevalent in Web 2.0 for reasons that I shall explain and explore later.

In the platform configuration shown, external innovators provide services that are consumed by the users of the platform for a fee, which could be collected by either party depending on the channels to market. However, the most common business model is some kind of revenue share between the platform owner and the external innovators. For example, if a telco provides an API for call control of some kind, perhaps allowing mobile users to call a party-line, then revenues from the minutes could be shared with the service integrator consuming the call-control API.

Examples of this kind of platform are:

1. Amazon marketplace
2. ITunes app store
3. Games consoles
4. Operator app stores

1.5.3 Data and Mash-Up Platforms

Figure 1.12 shows a different platform combination. On the left hand side, the telco is exposing data via APIs. External innovators on the right are using the APIs to request and consume data from the platform. The data could be anything, but is usually related to either the users of the platform (e.g. Customer data) or one of the key enablers of the platform, closely aligned to the core business, such as weather forecasting data from a weather forecasting venture. In the case of operators, a good example would be the location of customers.

Again, the telco goes about its business as usual. There is no special configuration or iteration of the core IT stack. Data is consumed without operator intervention, notwithstanding any data governance enforced by the API infrastructure put in place to expose the APIs in the first place. The innovators on the right hand side of the diagram have their own business, which doesn't have to be aligned at all with the core platform business. This is quite different to a platform play like games consoles where the innovators are narrowly confined to providing games per

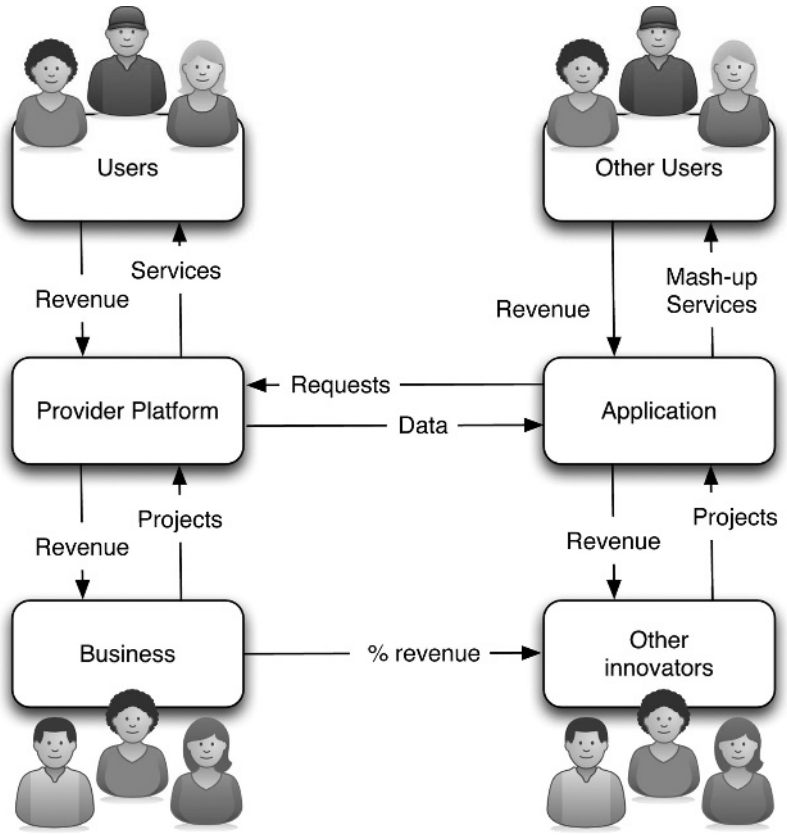

Figure 1.12 Mash-up platform configuration.

the business objectives of the platform provider (e.g. Sony). In this configuration, the platform could be something related to security services, for example, totally without alignment to the communications services of the platform provider.

The innovators build an application (or have an existing application) that requests data from the platform APIs and consumes it into the application itself, using it to deliver services or service features that add value to the application users. Note that the application users might not be platform users, although this is often not the case. The data consumed via the APIs is mashed with the data from the application itself, hence the term "Mash-Ups," which was a term coined to described the appearance of Google Map interface elements inside of other websites outside of the Google Website properties. However, there is no reason for interface mash-ups unless this is the only way that the API generates data (i.e. as a finished map). It might be that the API generates raw data that is consumed and then integrated into the application. The data might never be displayed at all, used only to control the business logic inside of the application.

There are various business models for these sorts of API configurations, but they generally fall into either direct or indirect revenue increments for the platform provider. In the case of

direct increments, it is likely that usage of the APIs is monetizable – the platform provider charges for usage of the API. There is no single business model for this, but low-friction is still important, so we expect to see the "freemium" model used where initial (or low) usage of the APIs is free with a charge levied later on, most likely for substantial usage of the API. This is a common model across the Web.

With indirect revenues, we expect to see more revenue generated from users of the platform as a result of the API being used. Google Ads is a great example. The more that ads are consumed via the API, the higher the chance that more users will click the ads, which causes the platform to levy charges to the users. These revenues are then shared back with the API consumer.

Examples of this platform approach are:

1. Google Ads
2. Amazon affiliates
3. Navteq Maps
4. O2 #Blue Service

1.5.4 Platform as a Service

The final pattern is shown in Figure 1.13.

This pattern is the logical conclusion of the platform idea, in that the platform itself becomes the service. Platforms are no longer an extension or strategic side-dish for the provider – platforms become the core business within their platform play. For the innovators using the platform, it becomes their exclusive means of delivering services to the end users, causing the innovators to host their applications on the platform.

In this model, the platform owner charges a fee for hosting the application, wrapping the hosting with a number of features that make the process of delivering services as easy and powerful as possible. The idea behind the Platform-as-a-Service (PaaS) model is that the application developer is freed of the various infrastructural overheads of delivering a service to users, allowing the innovators to focus on the central proposition that adds value to users (whereas the infrastructural components are largely invisible to the users).

The innovators bring their projects to life through the platform. The innovators construct the applications using their own tools and processes, or using tools provided by the platform. Some platform configurations might embed enough tools to design and deliver the service entirely, such as through the use of visual programming paradigms accessible via a Web browser. I have piloted such platform designs myself, allowing developers to invoke applications on the platform and then create the service by integrating a number of modules that call various APIs both on the platform and out on the Web. The platform orchestrates the tasks carried out by the various modules to deliver an on-demand service to the users.

An example would be a service that combines data from an online diary with location data in order to build an online booking service for a solo merchant, such as a plumber. The platform here also includes the various infrastructural components to allow for optimal delivery to a range of handsets and to integrate additional services like text messaging and click-to-call, should the merchant wish to provide the widest set of tools for booking new appointments with his or her customers.

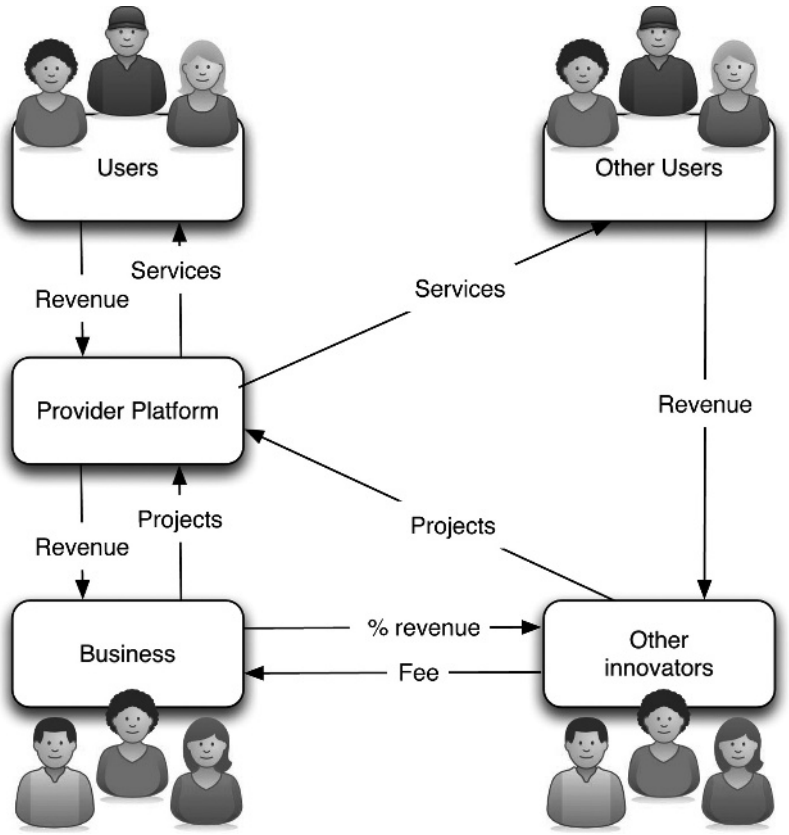

Figure 1.13 Platform as a service.

A typical business model for PaaS is to charge the innovators for the service. The model would include a monthly fee that scaled according to usage, which has a variety of possible quanta, including:

1. Number of applications hosted by the platform.
2. Number of users signed up to the applications.
3. Amount of usage of the applications (measured in all manner of ways).
4. Amount of infrastructural services being consumed by the applications – with various charges per infrastructural element.

The various chargeable infrastructural components might include:

1. Hosting and delivery of the service.
2. Charging and billing mechanisms.
3. Support mechanisms.
4. Analytics.

5. Personalization services.
6. Communications and messaging services.

Of course, the platform provider has the means to be flexible with the business model. An alternative is to share revenue with the innovators, which depends on how the revenue is collected.

In the case of a telco providing the platform, the dynamics of the business model will depend a lot on how the platform services assist directly with the core business versus being an entirely separate business operation and source of revenue. As the diagram shows, the platform can deliver services both to the platform owner's users and to the innovator's users. It might be that the platform owner sees an external service as being particularly attractive to its users and complimentary to its core business, in which case it might choose to enter a revenue share agreement if this enables a dynamic that leads to better service provision and uptake. In any case, the beauty of the platform approach is the elimination or reduction of any upfront capital costs, moving the costs to operational expenditure only. Moreover, the platform owner will usually provide a scalable pricing model with an attractive starting fee, thereby enabling a low-friction entry point, which is critical for platform adoption. As the service gains traction with users, usage goes up and the platform can deliver scalable infrastructure that grows with usage. The dynamics of scalable platform pricing are a crucial part of the PaaS story that has attracted so many application developers to various PaaS solutions.

As of the time of writing this book, there are signs of a new wave of PaaS solution that seem to be enabled by the ubiquity of underlying PaaS providers like Amazon Web Services (AWS). What AWS has done is to provide a very low friction platform on which innovators can build their own platform services, causing some commentators to think of AWS as something more like Infrastructure as a Service (IaaS), although such terms are not standardized. AWS has spawned an entirely new service industry where innovators have packaged common application components into their own PaaS offerings. This is a model that I have supported myself via the O2 Incubator program that I designed and ran for O2. One of the start-ups built a platform that aggregates and filters various information feeds across the Web (in particular social-network feeds). The platform gathers the channels into streams and then presents these back out via APIs. The advantage of the platform is that it removes a considerable processing and design burden from the application developer wishing to consume data from the social Web. Metaphorically, I imagined the service as a "Web switch" that switched social-Web conversations rather than voice conversations handled by a conventional telephony switch.

1.5.5 Do Platforms Work?

The answer is a resounding yes. The concept is not new. As Andrei Hagiu asserts in the title of his book, platforms have been the *invisible engines* powering vast business networks – ecosystems – that grow up around the platform, sucking in more users and innovators via network effects. The platform concept is not new to operators either, such as those who have ventured into the realms of enabling MVNOs. However, clearly operator business models are single-sided, dominated by a collection of revenues from the end users. All operators are aware of the vast potential hidden away in their IT stacks. The enticing notion of monetizing this potential has been discussed for some time and finally seems to be making its way into the

daylight, thanks mostly to the Web, which in the broadest sense has become the ultimate platform of platforms. This trend will only continue as the Web becomes more powerful, enabling new forms of data economics, many of which I shall be exploring throughout this book.

Whether or not platforms will be a success story for operators, it is still too early to tell, although I am strongly of the opinion that platforms need to become an integral part of operator futures. Many senior stake-holders remain sceptical of this, which is understandable. Stake-holders in telcos are from a different background and mindset than the Web-savvy entrepreneurs building new empires on the Web. I have been here before, many times. When I first set up a business in 1996 to offer Web applications to the telco industry, the Web was viewed with much suspicion and as a fad. Stake-holders have habitually repeated such reactions to all manner of Web-born ventures and patterns, like social networks and instant messaging and now APIs. On each occasion, they have lost out to disruptors who have nothing to lose in trying to build new platforms on the Web.

However, this time around I would caution telcos to take platforms more seriously than previous emergent Web patterns. This is because the platform model has accelerated so fast that it has almost become a hygiene factor in the delivery of any modern digital service. If it is ignored by telcos, they will find that non-traditional competitors eat their lunch in the emerging world of mash-ups and highly personalized digital services and connected experiences.

1.6 From Platforms to Ecosystems

It's not enough these days just to talk about platforms. With the acceleration of Web framework technologies and the open source movement, combined with the vast numbers of fixed and mobile internet users, it is increasingly easy to start a platform play. In fact, degrees of "platform-ness" have crept into many Web ventures – it is almost standard to offer some kind of API on which developers can extend the basic offering.

With so many platforms vying for attention, the name of the game has changed to ecosystems. You can think of this as supporting an entirely new industry around a platform, including developers, entrepreneurs, investors, evangelists, fans and all kinds of niche outcrops from the base, as shown in Figure 1.14.

Twitter is a great example. The platform has already attracted over 50,000 applications with all kinds of commercial and non-commercial intentions. In turn, these ventures have attracted venture capital into the mix, chasing lots of opportunities with an opportunity to scale. Many niche categories of application have emerged, such as social-marketing, sentiment analysis and so on. Chances of success might be remote in most cases, but too tempting to ignore overall. Once established, ecosystems can grow rapidly because too many players don't want to be left out.

Bloggers, journalists, even stars, have been attracted to the Twitter platform and its various offshoots. Many businesses have adopted Twitter to participate in the conversation about their brand and services. It has become a defacto way to follow what brands have to say about their wares.

Digital ecosystems, like natural ones, might become successful "by accident," but the various inflows leading to the tipping point can be encouraged and primed. Hence, it is important to have a sense of ecosystem in the strategic plan for some of the future operator models. This means paying attention to a much wider set of ingredients than just developers, service-enablers

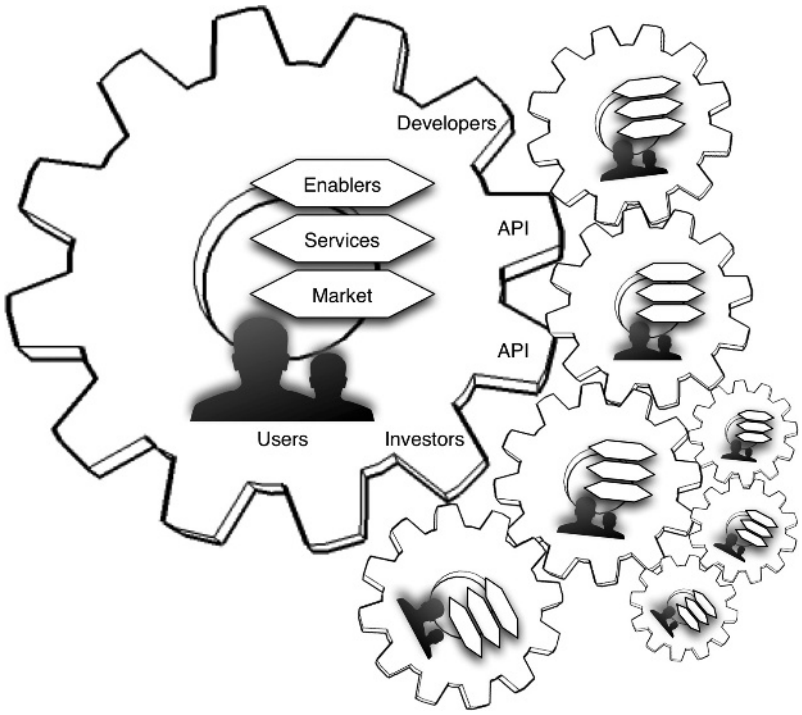

Figure 1.14 Ecosystems – a mutually supportive and competitive environment.

and APIs. You need evangelists, investors, start-ups, bloggers and a lot more besides to get that all important "echo chamber" effect that we see often in Silicon Valley. A platform is a lot bigger than just APIs and an ecosystem is even bigger still.

1.7 Where's the Value?

I'm not going to answer the "how do we make money from the Web?" question here. The title of this section is more a recognition that this question almost always gets asked inside telcos. I don't have the answer, nor is this book about giving the answer, at least not directly. But, it is important to know the right and wrong responses to this question from the outset, which is why I tackle it right now in the opening chapter.

In my experience, although this question is clearly important, the wrong conclusions are reached all too quickly, which I can explain within the context of what I have covered so far in this chapter, particularly the future models for operators (see Section 1.3 Six Models for Potential Operator Futures).

> **Lesson 1 – Money is where the users are –** That seems obvious, because it is. Yet, in this context it means that if we can't build platforms, services or applications that attract lots of users in interesting and new ways that address their problems, then we can't ask them for money. This is generally agreed. However, what many

folk in telcos still don't get is that delivering the initial service for free, or at least part of it, is often the cost of building a platform on the Web. There simply aren't that many examples of Web platforms that charged from day one and made a good profit, except in certain SME and enterprise categories. However, there are arguments both ways and I will return to the issue of charging from day one when I explore the patterns of start-ups (see How Web Start-ups Work).

Lesson 2 – Low-friction services are key – If it is possible to gets users to benefit from a service with minimal fuss, and drop an extra (small) charge on the bill in return, they might well go for it. The moment the service friction increases, to even a slight pinch of pain, all bets are off. The friction-value sensitivity curve has a very sharp inflexion.

Lesson 3 – Web is hygiene – To remain a relevant service in a world of increasing convergence and digitization, service delivery via the Web is essential and all that it entails in a post Web 2.0 world, which includes being socially-connected, context aware, open etc. In other words, the idea of becoming an "online business" is simply wrong-headed. Online is hygiene. Now the challenge is that born-on-the-Web companies generally do it much better. This needs to be fixed regardless of the money question.

Lesson 4 – Value is in the data – We have reached a point where the cost of storing and processing vast quantities of data has created new economies of scale. It pays to have a strategy that involves exploiting vast swathes of data. It pays to have a strategy that exploits potentially deep and complex number-crunching. Don't forget – it might be hard for you or I to contemplate complexity, but it isn't for a cluster of computers. There are entirely new opportunities to make money from handling large volumes of data.

Lesson 5 – The answer is in the method – This is probably the most important lesson, which is that we should accept from the outset that we don't yet have a business model, but we will set up our approach such that we maximize the chances of finding a viable business model. In other words, in Web start-up ventures (whether by a brand new company or an established one), the objective is to search for the business model as part of the initial phase of the venture. We suspend judgement about the model until we've actually built something that resonates with users and provides the answer to the money question. The secret to this approach is in the agility of the venture. The quicker and leaner we can operate to iterate through various service ideas, the better our chances of finding the business model before we either run out of money or patience. Being agile is key!

1.8 What Should We Build? It's Still About the Experience!

I still get asked this question time and time again – "what should I build?" Partly, it is that desire we all have to unearth a gem – to unlock the killer app. I always respond with the same advice, which has two parts:

1. Just build something and put it out there.
2. Focus on a compelling user experience.

Let's boil these down a little.

Firstly, it is essential to grasp, per the Eric Ries Lean Start-up Methodology (see Chapter 10) that the Web is still so young and its canvas so vast that we are constantly operating in the domain of the unknown. We don't know what will work, what won't, in terms of *new* ideas. This book will certainly outline a lot of the technological patterns that are behind things that work or appear to be working, so you're in the right place. However, one characteristic of these technologies and the Web frontier is that they move quickly. In the domain of the unknown, agility is king. You have to move fast, try things out, tune, adapt and keep moving, trying to find resonance with users before running out of cash.

But one could still be agile at delivering rubbish, so the focus must continue to be on the user experience. This message can't be said enough times because it is repeatedly ignored, even though it is startlingly obvious. We can talk about user experience (UX) in depth. It's a big topic. UX is almost always confused with user interface (UI), but let me be clear about this:

$$\text{UX} \neq \text{UI}$$

They are not the same thing. UI contributes to UX, but does not define it. I believe that many of my colleagues who are self-proclaimed UX experts also don't understand this topic in totality because UX is almost always seen as synonymous with visual design. It is not. UX is synonymous with utility.

BEFORE you start emailing me on this one, which I hope you do – I don't want to get into the semantics here. I'm not a UX expert, per se, although I have always seen my job, as someone who designs products and evangelizes various technologies, as generally being in the "user experience business." I just want to make a few simple points about this topic that seem to trip many of us up, especially in the telco world where they have historically seen UX as synonymous with handset usability.

Let's start with its most basic synonym, which is benefit. If the service has little or no benefit, then it doesn't have utility and it will be a poor user experience, no matter how much time and effort is invested in the design. This is bag-of-hammers-dumb obvious. Yet, the mistake made time and again is to define benefit in the mind of the product owner, or, more typically in large companies, inside a room of "intelligent people," which is usually a bunch of marketing guys armed with customer data.

In the Lean Start-Up methodology, the notion is to test the hypothesis with real users and real feedback, not conjecture and insistence on an idea because its owner has attachment. In the field of UX, it's often better to either discount one's own experiences, or at least to validate them. Unless you happen to be a product guru with a genuine feel for your users – and such people are rare – your tastes and product expressions are unlikely to resonate with the bulk of your users. Wherever possible, be guided by the users, not your instincts. Better still, be guided by the data obtained from real feedback and experience with your product.

It is also vital to appreciate that a lot of products and services can gain a foothold, even become wildly popular, despite a relatively poor design. This occurs when the product does

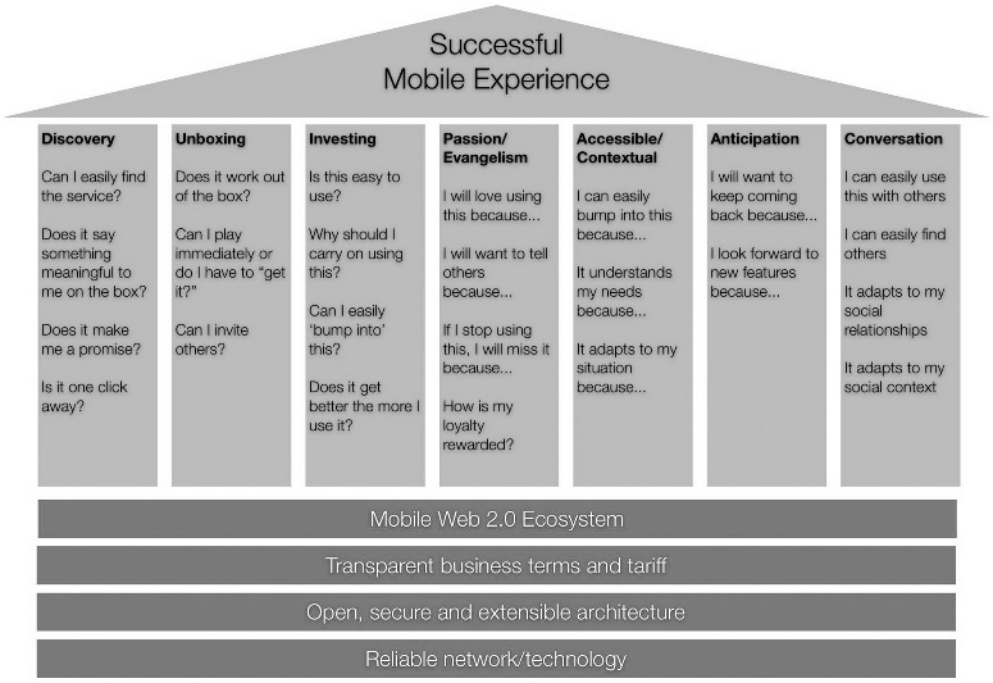

Figure 1.15 Components for a successful user experience.

an existing job but at a much lower price. When the price is lower, users will be more tolerant of product limitations, so long as – and especially if – they gain access to a new solution that was previously out of reach. Again, this is about utility. Because the user can do something previously unreachable or impossible, the experience will be satisfying.

The other failing of thinking of UX as UI, is that it misses overall context. Firstly, the UI is only one aspect of how a user experiences a product. For example, when one of your users tells another person, say a non-user, about your product, this is part of their experience of the product. When a user tries to discover the product, this is part of the experience. For these reasons, I came up with an example diagram to illustrate the various ways that UX can be approached in the total product experience, as shown in Figure 1.15.

The above diagram is only illustrative of the kinds of experiences that a user might encounter in using a product or service. It is deliberately in the first-person because of the way I use the diagram in workshops. However, in a wider setting, it would be matched with different user personas that we might envisage. We should not forget that in a platform model, there are two sets of experiences and personas to consider: the users and the developers. They are both important. UX should always be the starting point in our product, service and platform strategies. It has an important influence on the overall success regardless of underlying technologies and business models.

1.9 Summary

What this book will reveal to you is a collection of key technological patterns. I have outlined some of the key themes early in this chapter that will set the scene for much of what's to come:

- "Http://" is the dial tone of connected services. Combined with our ability to access IP from anywhere, Web software is "every-ware." In the telco world, we tend to think that transactions happen on the Web in IP, but the real vocabulary of Web innovation is software – widely available, flexible and powerful.
- Operator models will evolve, most of them to include greater Web-intimacy, if only because that's where digital lives are now being lived with more personal investment and attention than ever.
- The Web is the InterWeb – a collection of ecosystems. One of them is the "Social Web," which is underpinned by social graphs and their associated platforms and software elements. Big social networks have driven this trend and are now defining it. Operators need to regain relevance in this field. There are still plenty of opportunities.
- The name of the game is platforms. These are frequently hard to execute and need a lot of "developer love." Know that you're a platform and behave accordingly. Think low-friction.
- Beyond platforms are ecosystems – a much wider collection of actors and resources that make a platform successful. The best platforms will spawn into ecosystems, often by accident, but the rights seeds need to be sown – platforms are more than just exposing APIs.
- It's no longer viable to use the money questions as an excuse not to become an "online business," meaningless though that term is rapidly becoming, with so much of our world already online. In the fast-moving world of online, where business models aren't always as certain as in the telco world, a key competence is agility.
- Even with the best platform strategies and grasp of all the technologies in this book, we can never neglect the overall user experience. It's the starting point of everything that follows.