Growth and Innovation in Software Ecosystems[*]

(Work in Progress)

Geoffrey Parker	Lones Smith	Marshall Van Alstyne
Tulane	Wisconsin	Boston University/MIT
gparker@tulane.edu	lones@ssc.wisc.edu	mva@bu.edu

July 18, 2013

Abstract

This paper introduces and develops a new model of platform growth, inspired by the standard growth literature. At each moment in continuous time, the platform owner must choose (*i*) how much source code to open and make available to a developer community, (*ii*) how fast to absorb developer code into the platform, (*iii*) how much to tax developer output, and (*iv*) how much to invest in the platform itself. Without any public code, the control problem reduces to the growth model of Solow (1956).

Our platform model captures an infinitely recursive R&D ecosystem with innovation spillovers across developers. To optimize growth, the platform owner must internalize this recursive externality. Developers also innovate less the sooner they lose their code, but the platform grows faster with earlier code expropriation. The platform thus willingly privately contributes to the collective public good of open source code, as part of a dynamic profit-maximizing strategy to encourage taxable innovation.

We find that with the nonrival character of software, there are initially *increasing returns* to irreversibly releasing code into the public domain. Consistent with a variety of data, larger platforms should impose higher developer fees and taxes, expropriate less, and save more. We conclude by comparing the platform to a benevolent social planner.

[*]This paper is supported by National Science Foundation grant #0925004.

1 Introduction

A substantial and growing fraction of the economy is represented by software platforms — namely, closed economic ecosystems whose owner sustains their development, and who can set rules for opening platforms and for choosing when to bundle downstream applications into the platform. The platform owner faces a dynamic economic control problem with options and tradeoffs not jointly present in other economic settings. Notably, he can divest some of his platform code into the public domain and thereby profit from an external community of developers endowed with commercially valuable ideas that augment this code. For instance, while Myspace created its entire platform by itself, Facebook allowed outside developers to build new applications.[1] That Facebook has since proved a stunning business success and Myspace a catastrophic failure speaks to the critical role of played by developer firms in the platform viability.

We develop a parsimonious model of platform optimization of its software ecosystem that embodies several key features of the optimization they face and its social welfare implications: platform depreciation and re-investment, heterogenous developer ideas, developer taxes, code expropriation. In our key modeling innovation, the platform must choose how much of its code to open up for a heterogeneous continuum of small developers to build upon. Developers indexed by the quality of their product ideas. The platform owner becomes a landlord to developer firms that build and sell applications on its property. When a traditional landlord allocates land to one tenant, it is unavailable to another. But here all firms can productively encamp on the same ground, since the code is a nonrival good.

Since platform code depreciates rapidly, the platform must choose how fast to re-invest revenues into building the platform. These production and savings-investment decisions

[1] "We tried to create every feature in the world and said, 'O.K., we can do it, why should we let a third party do it?'" says (MySpace cofounder) DeWolfe. "We should have picked 5 to 10 key features that we totally focused on and let other people innovate on everything else." — Felix Gillette, *Business Week*, 6/27/2011.

resemble those in the standard exogenous growth paradigm in macroeconomics that we parallel (Solow, 1956). But we differ critically in having two legal dimensions of capital — privately-held and publicly-held platform code. For instance, modular programming is an example of public code that can serve as the bricks and mortar for future developers code (MacCormack et al. (2007)). We show that releasing code into the public initially benefits from increasing returns. For the more code that is released, the more developers are able to devise new and more imaginative applications.

In our continuous time setting, the platform constantly depreciates. Indeed, one should expect that code depreciates much faster than physical capital both because of constant attacks on its integrity (wear and tear), and given the rapidly changing technological frontier (rust and dust). The platform owner therefore must make a critical choice of how fast to replenish the public domain code. But this activity is by no means a free lunch. For the decision to release code into the public domain is an *irreversible* one. The platform must worry that an upstart can learn from his published code, and produce its own rival platform. Faced with potential competition from other platforms that may arise, we assume that the platform owner can only charge for the surplus value his private code confers upon consumers. This embodies a classic tradeoff that more openness helps future innovation at the cost of lower current profits (Besen and Farrell, 1994).

Our paper contributes to the dynamic R&D literature. For we allow two platform choices regarding the developer ecosystem: First, developers may see their revenues taxed by the platform — as with Apple's 30% rate on iPhone applications. We therefore model the choice of the platform developer tax rate. Second, unlike firms producing tangible standard rival goods, the platform owner may at some point copy or confiscate an application that adds value to the platform. We model this code expropriation simply as a choice of how long application producers may own their products. In practice, a platform owner often

creates a functionally equivalent application he then bundles into the platform.[2] This is analogous to a choice of patent length for R&D firms, and captures an intrinsic tradeoff: Bundling applications into the platform sooner reduces innovation incentives, but enriches the platform for future developers. This captures the essence of the *ex ante – ex post* tradeoff in technology markets. The platform eventually embeds arbitrarily many rounds of innovation sequentially layered on one another. Our paper tells a much richer story of R&D than usually told,[3] with profit and adoption tradeoffs in open versus closed code.[4]

The platform sponsor profits from the developer ecosystem in two ways — the tax rate on developer revenues and the finite proprietary period before expropriation. A higher tax rate or shorter proprietary period helps the platform, but reduces developer profits, and thereby this shrinks the developer ecosystem. The platform sponsor selects a positive tax rate and positive expropriation rate that equate his dynamic tradeoff with the developers.

We model the platform environment as a closed economy. Faced with a depreciating code base, the platform sponsor is a social planner who must directly reinvest in its maintenance. He must balance current against future consumption at the margin. As with the macroeconomic growth literature, we compute the golden rule solution that maximizes the platform's steady state value. We ask how this solution varies with the basic parameters. We show, for instance, intuitive result that the platform owner should more heavily re-invest in the platform when the depreciation rate is greater. Less intuitively, we argue that platforms serving larger markets should impose smaller taxes on developers, and instead rely more

[2]This has happened many times — for instance, Microsoft saw the vital importance of web browsing and released Internet Explorer with Windows 95. When Microsoft bundled web browsing, instant messaging, streaming media multi-threading, and disk compression into Windows — all features originally developed by downstream developers — venture capital firms reduced funding any startups whose products might be bundled into the operating system.

[3]The leading papers here Green and Scotchmer (1995) and Chang (1995) explore two stage innovation.

[4]Closest in spirit to our paper is Boudreau and Hagiu (2009). They show that when discoveries sequentially build on one another, patent protection is unwise, and that inventors may find higher profits in a world with competition and imitation. This idea emerges in our setting, but without the same developers moving twice.

heavily on code expropriation. This finding partially makes sense of the different strategies employed by the Windows and iPhone platforms.

Our point of departure from the growth literature mirrors the knowledge spillover growth model of Romer (1986). But whereas he focused on the positive external effect of new knowledge that "cannot be perfectly patented or kept secret", we explicitly model the public domain code as a capital stock.

WHITHER CLOUD COMPUTING? We allow the platform to write contracts with developers that includes code expropriation. In a world with cloud computing, the platform would not own the domain, and would lose this lever of power. We see in §5 would lose its dominant source of platform growth.

We next analyze in sequence the platform model of private and public code, explore optimization with the platform ecosystem, and then the platform dynamics. Finally, we turn to the twin platform and planner's problems.

2 Motivation

Depreciation is much more important than holds for standard physical capital (eg. 15%) in the macroeconomy for two reasons. First, there may be significant obsolescence, as the technological environment is rapidly changing; therefore, software must be quickly re-configured, possibly with new functionality added. Second, the platform software environment is prone to attack by hackers, and this ongoing war requires constant repairs and defensive adaptations. Microsoft updates for Windows operating systems, for instance, often arrive more than once a week in response to the latest discovered vulnerabilities.

platform code can be re-used, given software modularity

The platform owner also takes a "piece of the action" from the applications firms — for

instance, Apple taxes 30% of developers revenues for the iPhone and zero for Mac.[5]

The platform owners can expropriate any piece of software it deems mission critical. What it does in practice is make a take it or leave it offer, and threaten replication. We will treat this as "eminent domain".

We assume that the platform managing the ecosystem invests in its code and and expropriates code. This captures many real life features. Platforms often acquire companies to secure their code. But behind the scene is the threat that the platform will replicate the code.

- Apple acquired NeXT in 1997, it acquired the core for its new MAC OS kernel.
- Mac: eMagic became Garageband
- Soundjam became iTunes

As long as the platform offers the developer a discounted price, this is best modeled as expropriation. If the platform offered the developer full value, then it is essentially investment in the platform.

Really, platform code is not all the same.

- Developer library is published.
- Video game platforms like XBox typically charge licensing fees from $3 to $10.

Our code dynamics describe an open source ecosystem like Lynux, but our platform optimization is inapplicable there.

THE ECOSYSTEM. Steve Jobs boasted at 2010 Apple World Wide Developers Conference of 225,000 iPhone and iPod touch apps in the App Store, and an excess of $1 billion of tax revenues from developers in the App store. The number of developers for the Windows platform swamps this number.

[5]Windows charges 0%; Salesforce.com charges 30%; Amazon Kindle charges 70%.

3 The Platform Model

Growth theory explicitly models production as a function of labor and capital, and explores the capital dynamics given exogenous depreciation and investment. We develop a related dynamic model of a platform optimization with a two dimensional capital stock: a mass $K > 0$ of *private code* known only by the platform owner, and a mass $L > 0$ of *public code* that can be built on by others. The distinction is at the heart of our contribution.[6]

In the spirit of growth models, we assume a concave production function of the total platform code $K + L$. More specifically, total code $K + L$ is worth $\psi(K + L)^\alpha$ to consumers, where $0 < \alpha < 1$. Think of $\psi > 0$ as a measure of the *economic size* of the platform market.

Our public code presents the key substantive economic difference from growth models. A start-up firm could presumably bundle the public code itself, and create a free limited platform of consumption value L^α. We thus assume that the platform owner is constrained by *potential competition* from future platforms that may arise by enterprising upstarts building on this public code. To best capture this, we assume that the platform owner can only profit from the value-added conferred by his private code. We do not model the platform pricing decision except to assume that it extracts a platform rental fee equal to the annuity value of the difference $\psi(K + L)^\alpha - \psi L^\alpha$.

The public and private code bases evolve with the passage of time. We suppose that code erodes at a constant *depreciation rate* $\delta > 0$. Balancing this, private code has two engines of growth. First, the platform owner produces new code. We assume without loss of generality that it only adds to the private code base, and then sustains the public code by "divesting" private code into the public domain. To this end, let $\pi > 0$ be the *publication rate* of private code into the public domain. Next, the platform owner maintains private code by re-investment. Analogous to the Solow growth model, we assume that the platform

[6] We suppress the time argument for K, L and all other variables in our model for notational ease.

owner pays for investment by saving a fraction $0 < s < 1$ of his platform revenues.

For the second source of growth, we allow the platform to exploit his property rights on the developers, since they are building on the platform domain (K, L). We first allow the platform owner to write contracts with the legal spirit of "eminent domain" provisions: An application developer has only a limited duration monopoly on its product, after which its code is folded into the private platform code. Reflecting this, we assume that the platform owner *expropriates* developer code into the private code base at some rate $\chi > 0$. Such a stochastic exclusionary period induces a stationary model. Secondly, we allow the platform owner to seize a "piece of the action" from the applications firms before expropriation. We venture that the platform chooses a *tax rate* $0 < \tau < 1$ on developer revenues.

Altogether, the platform owner chooses parameters χ, π, s, τ to maximize the present value of the revenue flow from direct sales, less re-investment costs, and developer taxes:

$$(1 - s)\psi[(K + L)^{\alpha} - L^{\alpha}] + \tau(\text{developer ecosystem revenues}) \tag{1}$$

We now explore the developer ecosystem, thereby fleshing out the latter term.

4 The Developer Ecosystem

The developer ecosystem sustains the external returns to public capital. In a modeling assumption consistent with this myriad of firms, we assume free entry from a heterogeneous continuum of potential developers.

A. Individual Developers. The continuum assumption embodies a lack of market power by developers. The source of increasing returns in this world is the ever-welling pool of external ideas. Developers differ by their ideas $x > 0$, where we envision that an idea simply scales up the value of the developer code. While developers live in a dynamic world, their

decision formally reduces to a static optimization. We do not model demand, and instead work in a reduced form setting — as if assuming infinitely elastic demand — where the production function translates directly into revenue. It is also in the spirit of the Solow growth model, in which the planner simply cares about productivity. This is consistent with our assumption of no market power, and our laser focus on how the ecosystem contributes software to the platform. We assume that by using public code $L > 0$, any application firm with idea x can produce code of *present value*

$$v(x, m) \equiv Axm^{\sigma}L^{\xi} \tag{2}$$

by using a variable input m, costing $wm + \kappa$. We might think of the variable input m as hours of engineers, for instance, while the fixed cost $\kappa > 0$ includes the one-shot opportunity costs of doing the start-up, and in particular, embeds any platform-imposed developer fee. Costs are measured upfront in current dollars — as if the developer first rents its facility and hires its engineers, and afterwards sells its product. The coefficient $A > 0$ is a technology productivity parameter, measuring the ease of platform code re-use. It is higher, for instance, with greater software modularity of any applications. Since software engineers and platform code are highly specialized inputs, we assume *diminishing returns to scale*, so that scaling up a developer firm's size increases its profits less than proportionately:

$$\sigma + \xi < 1 \tag{3}$$

Instead, the only source of increasing returns in our model is more and better ideas, which is the focus of this model. To this end, we assume an inflow of potential new developers. Of these, the mass of ideas x has density $g(x) = \mu\beta x^{-\beta-1}$ on $[1, \infty)$, where $\beta > 0$. Intuitively,

ideas $x \leq 1$ are worthless and so ignored.[7] Equivalently, there is a total mass $\mu x^{-\beta}$ of new potential developer ideas above any level $x \geq 1$. The tail size determines the level of the external returns: smaller β corresponds to a thicker developer idea tail. Not only is the mass of ideas finite, but we shall soon see that the tail must vanish much faster, with $\beta > 1$.[8]

Platform code reabsorption creates a dynamic pricing drag on developers. For they cannot sell code of consumer value $v(x, m)$ for full-price, because consumers can simply wait until it is rebundled into the platform. We assume the developers set a price $p(x, m)$ that leaves consumers indifferent about waiting, and buying immediately. Intuitively, a greater expropriation rate blunts a firm's ability to charge for its applications. Given the expropriation rate χ, consumers expect to secure a fraction ρ of developer value by waiting, where

$$1 - \rho \equiv \int_0^\infty \chi e^{-\chi t} e^{-rt} dt = \frac{\chi}{r + \chi} \tag{4}$$

In other words, the market price $p(x, m)$ of developer x solves the consumer indifference equation $v(x, m) - p(x, m) = (1 - \rho)v(x, m)$, namely, $p(x, m) = \rho v(x, m)$. This creates a curious inversion for the value of owning developer applications: *Unlike for normal durable goods, a lower interest rate lowers prices, rather than raises them.* For consumers enjoy a greater future outside option, and so are more patient; they are willing to pay less for any given application. Since the platform also values the developer revenues indirectly through its taxation, its optimization on χ and τ will take into account this interaction.

With this device, we hereby reduce the developer optimization to a static one, in which they can secure a price $p(x, m)$. Since each developer only retains a fraction $1 - \tau$ of its

[7] The left tail under this mass density has infinite area when $\beta \geq 1$, which we will need; truncating it is essential in order to preclude a divergent dynamical system in §5, in which an exploding public capital stock is fed by an exploding mass of very poor developer ideas.

[8] This model subsumes the case in which ideas more than proportionately scale the output, say by $y = x^\zeta$ with $\zeta > 1$. For in this case, simply use our model with the idea density parameter β/ζ. But this logic reveals that β encodes production considerations too.

revenues, the expected present value of profits for one such firm with an idea x is then:

$$(1-\tau)p(x,m) - wm - \kappa = \rho(1-\tau)Axm^{\sigma}L^{\xi} - wm - \kappa \tag{5}$$

given the consumer value expression (2). The optimal input level $m^*(x)$ obeys the first order condition:

$$wm^*(x) = \rho\sigma(1-\tau)Axm^*(x)^{\sigma}L^{\xi} \equiv \rho\sigma(1-\tau)v(x,m^*(x)) \tag{6}$$

Substituting this factor demand function into (5), the developer with idea $x \geq 1$ earns profits:

$$wm^*(x)(1-\sigma)/\sigma - \kappa = w\left[\rho\sigma(1-\tau)AxL^{\xi}/w\right]^{1/(1-\sigma)}(1-\sigma)/\sigma - \kappa \tag{7}$$

Combining (6) and (7), his optimized flow software creation value is

$$v(x,m^*(x)) = \frac{\left[\rho\sigma(1-\tau)Ax/w\right]^{1/(1-\sigma)}}{\rho\sigma(1-\tau)}L^{\xi/(1-\sigma)}$$

Those firms with the best ideas $x \geq \underline{x}$ enter, and all but the marginal one earns positive profits. This yields the expression for the *marginal firm* with idea $\underline{x}$ earning zero profits:

$$\underline{x} = \left(\frac{\kappa\sigma}{w(1-\sigma)}\right)^{1-\sigma}\frac{w}{\rho\sigma A(1-\tau)L^{\xi}} \tag{8}$$

provided this expression yields $\underline{x} \geq 1$. At some point, once the public capital stock is large enough, we get $\underline{x} = 1$. Specifically, this happens when $L = \bar{L}$, where

$$\bar{L}^{\xi} = \left(\frac{\kappa\sigma}{w(1-\sigma)}\right)^{1-\sigma}\frac{w}{\rho\sigma A(1-\tau)} \tag{9}$$

The marginal firm determines the size of the ecosystem. Observe that for public code levels $L < \bar{L}$, entry into the developer pool is discouraged by higher costs κ or w or taxes τ, or code

expropriation (via the parameter ρ, in light of (4)), and encouraged by more public code L, and productivity A, the interest rate r (also via (4)). Finally, the ecosystem size obviously varies inversely to the marginal firm $\underline{x}$, and thus moves oppositely to each parameter.

B. The Whole Ecosystem. Assume that $L < \bar{L}$. Given optimal entry into the ecosystem, integrating across all active developers, the *gross flow value* of total code produced at each moment in time equals:

$$\mu \int_{\underline{x}}^{\infty} v(x, m^*(x)) \beta x^{-\beta-1} dx \quad = \quad \mu\beta \frac{\left[\rho\sigma(1-\tau)AL^{\xi}/w\right]^{1/(1-\sigma)}}{\rho\sigma(1-\tau)} \frac{\underline{x}^{1/(1-\sigma)-\beta}}{\beta - 1/(1-\sigma)} \equiv \Delta L^{\gamma}$$

where $\gamma \equiv \xi\beta$, and the constant D fully reflects parameters of the technology and developers:

$$\Delta = \frac{\mu\beta A^{\beta}(1-\tau)^{\beta-1}\rho^{\beta-1}}{\beta - 1/(1-\sigma)} \sigma^{\beta\sigma} w^{-1-\beta\sigma} \left(\frac{1-\sigma}{\kappa}\right)^{(1-\sigma)\beta-1} \tag{10}$$

The developer diminishing returns assumption (3) yields a bound on the returns to public code:

$$\gamma = \xi\beta < (1-\sigma)\beta$$

When the public code L exceeds the threshold $\bar{L}$, the marginal developer firm idea is constant at 1, and so by similar logic, the gross flow value of total code produced equals

$$\mu \int_{1}^{\infty} v(x, m^*(x)) \beta x^{-\beta-1} dx = \mu \int_{1}^{\infty} \frac{\left[\rho\sigma(1-\tau)Ax/w\right]^{1/(1-\sigma)}}{\rho\sigma(1-\tau)} L^{\xi/(1-\sigma)} \beta x^{-\beta-1} dx \equiv DL^{g} \tag{11}$$

where $g \equiv \xi/(1-\sigma)$. Next, the lead constant is

$$D = \frac{\mu\beta}{\beta - 1/(1-\sigma)} \left[\rho\sigma(1-\tau)\right]^{\sigma/(1-\sigma)} (A/w)^{1/(1-\sigma)} \tag{12}$$

For instance, the developer fixed cost κ does not affect the production of any of the firms,

since they are all inframarginal. As a result, ecosystem code is invariant to κ when $L \geq \bar{L}$.

In light of (10) and (12), the returns to variable inputs m cannot be too large relative to the size of the developers idea tail, or else the ecosystem value explodes. Namely:

Lemma 1 *The value of the ecosystem entry is finite if and only if* $(1-\sigma)\beta > 1$.

There may initially be increasing returns to public code for all $L \leq \bar{L}$, for $\gamma = \xi\beta$ possibly exceeds one. But the returns to public code are smaller for large public code levels $L \geq \bar{L}$, since $g = \xi/(1-\sigma) < \xi\beta = \gamma$ by Lemma 1. In fact, since $g = \xi/(1-\sigma) < 1$ by diminishing returns (3), *the platform ecosystem* (11) *has eventual diminishing returns to public code.*

Equally well, the *average present value of developer profits* equals the inflow of returns less the marginal and fixed costs. Exploiting our expressions for the marginal firm in (8) and the constant Δ in (10), the platform profits in (7) equal:

$$\text{profits} = \begin{cases} (1-\sigma)\Delta L^{\gamma} - \mu\kappa\underline{x}^{-\beta} = [(1-\sigma) - (1-\sigma-1/\beta)(1-\tau)\rho w]\Delta L^{\gamma} & L < \bar{L} \\ (1-\sigma)DL^{g} - \mu\kappa & L \geq \bar{L} \end{cases} \tag{13}$$

5 Public and Private Code Capital Stock Dynamics

The ecosystem analysis takes as given the public code level L. We now turn to the dynamic optimization facing the platform, and derive this critical capital stock. Given the assumed depreciation and code publication rates, the public code admits the law of motion:

$$\dot{L} = -\delta L + \pi K \tag{14}$$

Easily, we see that the public code stock would evaporate absent any private code publication.

Likewise, the private code base evolves according to the law of motion:

$$\dot{K} = f(K,L) \equiv -(\delta+\pi)K + s\psi[(K+L)^{\alpha} - L^{\alpha}] + \Delta L^{\gamma} \quad \text{for } L < \bar{L} \tag{15}$$

$$\dot{K} = h(K,L) \equiv -(\delta+\pi)K + s\psi[(K+L)^{\alpha} - L^{\alpha}] + DL^{g} \quad \text{for } L \geq \bar{L} \tag{16}$$

where $\bar{L}$ is given by (9). The first term in (15) and (16) includes the losses from depreciation of old code, and divestiture into the public code base. The middle term of captures the new private code due to savings re-investment. Observe that since $0 < \alpha < 1$, the difference

$$(L+K)^{\alpha} - L^{\alpha} = L^{\alpha}[(1+K/L)^{\alpha} - 1] \approx L^{\alpha}\alpha(K/L) = \alpha K L^{\alpha-1} \to 0 \quad \text{as} \quad L \to \infty \tag{17}$$

The last term in (15) and (16) is the developer code expropriation into private code.[9]

Corollary 1 *The platform code from reinvestment vanishes as public code L explodes, and so eventually, the platform code is maintained almost wholly by developer code expropriation.*

This has important implications for cloud computing. For this structure usually escapes the platform property rights, and thereby the latitude to expropriate. In other words, the only source of re-investment in a cloud platform structure is the vanishing revenues from reinvestment. This greatly limits the size of sustainable cloud platform structures. Unlike with standard platforms, monopoly is therefore a prediction for cloud computing.

Dynamical systems are often solved by understanding their resting points, and ours is no exception. In fact, we shall perform all of our analysis in steady state. Our two dimensional system enriches the analysis. Figure 1 plots the level sets of (14)–(16). By (17), the stationary curve $\dot{K} = 0$ curve at large L approaches the concave curve $(\delta+\pi)K = DL^{g}$. The diminishing returns intuitively suggests that a stable stationary fixed point exists, for similar reasons as

[9]In steady-state, developer code available to a platform accumulates over time comes from an infinite pre-history. Computing the total stock of code requires dividing the coefficient D by χ, since the expected longevity of any developer code before expropriation is $1/\chi$. Scaled by χ, this is the flow of expropriated code.

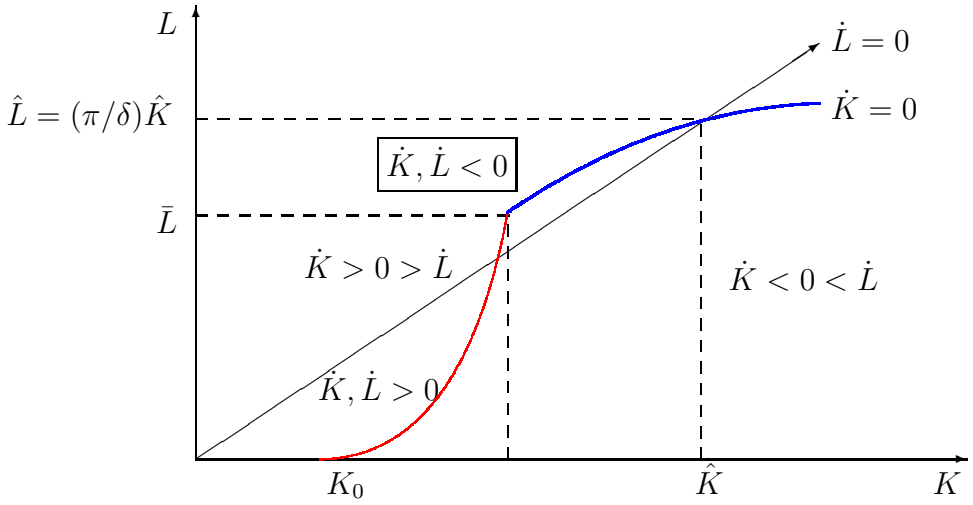

Figure 1: **Dynamical System.** This reflects how marginal returns to public code are initially increasing (thin red curve), but eventually diminishing (thick blue curve). The fixed point for private and public code levels exists provided $\underline{K} \leq \bar{K} \equiv (\delta/\pi)\bar{L}$. This fixed point is stable under the adjustment dynamics (14) and (15)–(16). The inequalities in the boxes capture the dynamics in the adjacent bounded areas.

in the Solow growth model. But if the diminishing returns kick in too early, then a crossing point might not obtain in Figure 1. For existence, it is sufficient, eg., that the increasing returns persists until the $\dot{K} = 0$ curve exceeds the locus (14) where the ratio of public and private capital stocks obeys $L/K = \pi/\delta$. In other words, it suffices that $f(\delta\bar{L}/\pi, \bar{L}) < 0$:

$$(\delta + \pi)(\delta/\pi)\bar{L} > \Delta\bar{L}^{\gamma} + s\psi[((\delta/\pi) + 1)^{\alpha} - 1]\bar{L}^{\alpha} \tag{18}$$

Since the constant Δ and the threshold $\bar{L}$ are independent of s, π, δ, this inequality holds for small enough savings rates s and code publication rates π, since $0 < \alpha < 1$. This is intuitive, for these assumptions jointly ensure that the private capital stock shrinks, $\dot{K} < 0$, and thus the dynamics lie right of the $\dot{K} = 0$ locus.

Altogether, for some public capital stock $\hat{L} > \bar{L}$, there is a unique intersection of $\dot{L} = 0$

and the diminishing returns locus $\dot{K} = h(K, L) = 0$ in (16). Here:

$$(\delta + \pi)(\delta/\pi)\hat{L} = D\hat{L}^g + s\psi[((\delta/\pi) + 1)^\alpha - 1]\hat{L}^\alpha \tag{19}$$

Theorem 1 (Unique Steady-State Code) *For small enough savings rates s and code publication rates π, there exists a unique positive stationary public and private code base pair $(\hat{K}, \hat{L})$ solving* (14) *and* (16)*; furthermore, it is stable for the dynamic adjustment process.*

Proof: The $\dot{K} = 0$ locus hits the horizontal axis $L = 0$ at the positive solution K_0 of

$$(\delta + \pi)K_0 = s\psi K_0^\alpha \quad \Rightarrow \quad K_0^{1-\alpha} = \frac{s\psi}{\delta + \pi} \tag{20}$$

Substituting $\hat{L} = (\pi/\delta)\hat{K}$, the fixed point of (19) can be rewritten in terms of $\hat{K}$:

$$\hat{K} = \frac{D}{\delta + \pi}(\hat{L}/\hat{K})^g \hat{K}^g + \frac{s\psi}{\delta + \pi}\left((1 + \hat{L}/\hat{K})^\alpha \hat{K}^\alpha - (\hat{L}/\hat{K})^\alpha \hat{K}^\alpha\right) \equiv \Upsilon \hat{K}^g + \Sigma \hat{K}^\alpha \tag{21}$$

for the developer and savings constants:

$$\begin{aligned} \Upsilon &\equiv \frac{(\pi/\delta)^g}{\delta + \pi} \cdot \frac{\mu\beta}{\beta - 1/(1-\sigma)} \left[\rho\sigma(1-\tau)\right]^{\sigma/(1-\sigma)} (A/w)^{1/(1-\sigma)} \\ \Sigma &\equiv \frac{s\psi}{\delta + \pi}\left((1 + \pi/\delta)^\alpha - (\pi/\delta)^\alpha\right) \end{aligned}$$

There is a unique solution to $\hat{K} = \Upsilon\hat{K}^g + \Sigma\hat{K}^\alpha$, as seen in Figure 2. Since it is a downcrossing, the fixed point is stable, as usual. QED

Theorem 2 (Comparative Statics) *The stationary public and private code bases $\hat{K}, \hat{L}$ both rise in the technology parameters A and α, market size ψ, savings rate s, and code expropriation rate χ; they fall in the tax rate τ, worker wages w, the interest rate r, and the depreciation rate δ.*

Proof: The graph in Figure 2 makes it clear that any parametric change that raises Υ or Σ pushes up the map $K \mapsto \Upsilon K^g + \Sigma K^\alpha$, and thereby inflates the fixed point capital $\bar{K}$.

- The developer constant Υ rises in A and ρ (and thus χ), falls in τ, w, r, δ, and is invariant to α, s, κ, ψ. Finally, Υ unambiguously rises in π for small π, but in general the effect of π is ambiguous.

- The constant Σ rises in α, s and ψ, falls in π, and is invariant to $A, \chi, r, \sigma, \tau, \kappa, w$.[10]

Comparative statics for the steady-state public code $\hat{L} = (\pi/\delta)\hat{K}$ follow since all shifts reinforce each other, except for the publication rates π. QED

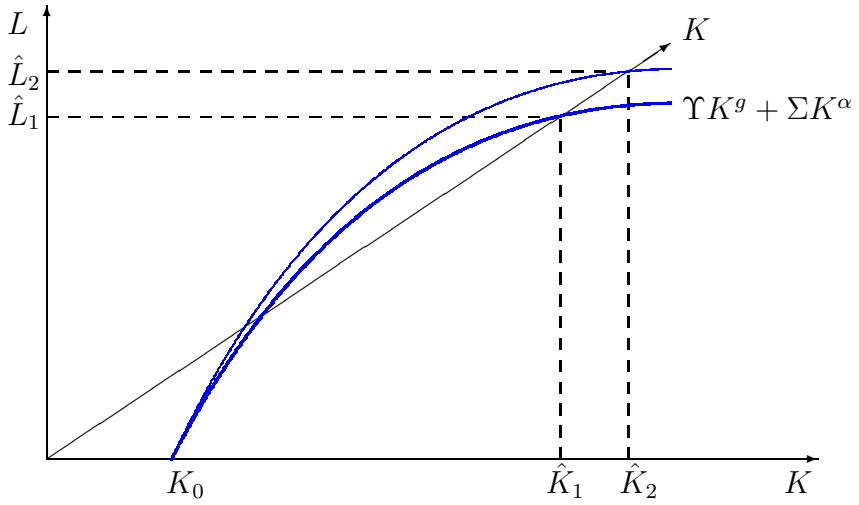

Figure 2: **Comparative Statics.** This depicts how the unique stable steady-state private code capital $\hat{K}$ that solves (21) rises from $\hat{K}_1$ to $\hat{K}_2$ as either Σ or Υ rises. The logic safely ignores the increasing returns portion.

6 The Platform Steady-State

The *Golden Rule* (Phelps, 1961; Ramsey, 1928) is the 4-tuple of savings rate s, tax rate τ, code publication rate π, and expropriation rate χ that maximizes steady-state revenues (1),

[10] Define $x = \pi/\delta$. Then comparative statics in δ require signing the derivative of $[x/(1+x)][(x+1)^\alpha - x^\alpha]$. The first term increases in x, and the second falls in x.

subject to steady-state in (14) and (16). We can make this an unconstrained optimization, substituting for the steady state code levels $\hat{K}, \hat{L}$ from Theorem 2. The steady-state value sums the revenues that are not reinvested in the platform and the total taxes from developers.

$$V(s, \tau, \pi, \chi) = (1 - s)\psi \hat{K}^{\alpha}[(1 + \pi/\delta)^{\alpha} - (\pi/\delta)^{\alpha}] + \tau\Delta(\pi/\delta)^{\gamma}\hat{K}^{g} \tag{22}$$

using the implicit definition for $\hat{K}$ in (21). We maximize this with respect to s, τ, π, and χ, and find positive derivatives at zero for these control variables. Thus:

Theorem 3 (The Platform Solution) *The Golden Rule entails a savings rate* $0 < s < 1$*, tax rate* $0 < \tau < 1$*, code expropriation rate* $\chi > 0$*, and code publication rate* $\pi > 0$*.*

Observe the simple implication of the positive code publication rate $\pi > 0$. Public code publication clearly has the flavor of the private provision of a public good. But unlike (Bergstrom et al., 1985), it is whole driven by the incentives for dynamic innovation. Indeed, the social value of the ecosystem owes to the code publication. *The platform initially secures higher geometric returns from the developer ecosystem production on its public code than its internal rate, of value* $(\Delta/\chi)L^{\xi\beta}$*, where* $\xi\beta > \alpha$*.* This source of additional returns, above and beyond the platform's own coefficient α, owes both to the expansion of existing developers and the entry of new ones. Eventually, in the region where the platform operates, its source of additional returns owes solely to the expansion of existing developers, and the returns are of the order $L^{\xi/(1-\sigma)}$.

We now venture some comparative statics predictions of the steady-state of the model.

Theorem 4 (Larger Platforms) *As the market size* ψ *rises, the golden-rule optimal savings rate* s *and tax rate* τ *fall, and the code expropriation rate* χ *rises.*

Proof: According to the method of monotone comparative statics, any optimized control variable x^* rises in a parameter ℓ if $V_{x\ell} > 0$ whenever $V_x = 0$ (Topkis, 1998). In other words,

we must sign the cross partial just when the first order condition holds. Changing the order of differentiation, and exploiting $V_x = 0$, we get

$$V_{x\psi} = V_{\psi x} = \left((1-s)\bar{K}^{\alpha}[(1+\pi/\delta)^{\alpha} - (\pi/\delta)^{\alpha}]\right)_x = -\left(\tau\Delta(\pi/\delta)^{\gamma}\bar{K}^{g}\right)_x \tag{23}$$

Now, $\bar{K}$ rises in s by Theorem 2, while D is constant in it. Hence, $V_s = 0$ implies that $V_{s\psi} < 0$, and so the savings rate s rises in ψ. Next, $\bar{K}$ falls in τ by Theorem 2. Thus, $V_\tau = 0$ implies that $V_{\tau\psi} < 0$, and so the tax rate τ falls in ψ. The same logic applies for the developer fees κ and the code expropriation rate χ. QED

Consistent with this result, Microsoft's XBox platform has yet to make a profit, having "saved" much more than it has earned in platform sales and developer taxes.

7 Welfare Analysis

The private code stock is the only means of either growth or replenishment of the public code stock. Additionally, this offers profits for the profit-maximizing platform, and this marginal additional incentive to build the private code stock is no longer present for a true social planner. So we suspect that the planner would choose a higher savings rate and/or publication.

The social planner values all the profits of the developers, and not just the tax revenues (1), as well as the entire consumer surplus. But the planner ignores the platform transfer from consumers, and developer taxes. In other words, the planner's steady-state value function is

$$W = \psi(\bar{K} + \bar{L})^{\alpha} + \Delta\bar{L}^{\gamma}/\beta = \psi\bar{K}^{\alpha}(1+\pi/\delta)^{\alpha} + \Delta\bar{K}^{\gamma}(\pi/\delta)^{\gamma}/\beta$$

hereby building on the expression (13) for total developer profits. How how the public code figures so much more prominently in the social planner's objective function than in the platform owner's. In other words, at the margin, those controls that inflate the public code will be larger. On the other hand, the planner does not care about developer taxes.

Theorem 5 (Social Planner) *Compared to the platform owner, the social planner would choose a zero tax rate τ, a larger public code publication rate π, and would expropriate faster.*

In other words, even though the platform owner is privately providing the public good, he is not doing so to the extent that a benevolent social planner would desire.

8 Conclusion

This paper proposes a dynamic model of platform growth reflecting the fact that platforms may construct closed ecosystems of firms that build on publicly available code. Our model takes inspiration from the growth literature in macroeconomics. We explain the logic of this platform strategy, and derive how the platform should optimize its treatment of developers.

A APPENDIX: OMITTED PROOFS

A.1 Proof of Theorem 3

To show $x^* > 0$, it suffices that $V_x > 0$ when evaluated at $x = 0$. For the controls $x = s, \chi, \pi$, this follows from Theorem 2 — recalling the properties of Δ derived in its proof, and that $[(1+\pi/\delta)^\alpha - (\pi/\delta)^\alpha]$ is second order in π near 0. Consider next the control $x = \tau$. For τ near 0, the fall in Δ is of smaller order than τ, since $\beta > 2$; therefore, the last term $\Delta(\pi/\delta)^\gamma \hat{K}^\gamma$ in the derivative of (22) at least initially dominates the fall in $\hat{K}$. So $V_\tau > 0$.

To see that any control obeys $x^* < 1$, we need $V_x < 0$ when evaluated at $x = 1$. For the control $x = s$, this follows from the same logic as did $V_\tau > 0$ near $\tau = 0$. For the control $x = \tau$, we have $\Delta = 0$ when $\tau = 1$. For a small decrease in τ from $\tau = 1$, the last product term in (22) rises, since both $\hat{K}$ and Δ rise, by Theorem 2 and its proof. QED

References

Bergstrom, T., L. Blume, and H. R. Varian (1985). On the private provision of public goods. *Journal of Public Economics 29*(1), 25–49.

Besen, S. and J. Farrell (1994). Choosing how to compete: Strategies and tactics in standardization. *The Journal of Economic Perspectives 8*(2), 117–131.

Boudreau, K. and A. Hagiu (2009). *Platform rules: Multi-sided platforms as regulators*, Chapter 7, pp. 163–189. Edward Elgar Publishing Limited, Cheltenham, UK.

Chang, H. (1995). Patent scope, antitrust policy, and cumulative innovation. *The RAND Journal of Economics 26*(1), 34–57.

Green, J. and S. Scotchmer (1995). On the division of profit between sequential innovators. *RAND Journal of Economics 26*(1), 20–33.

MacCormack, A., J. Rusnak, and C. Baldwin (2007). The impact of component modularity on design evolution: Evidence from the software industry. Harvard Business School Technology and Operations Mgt. Unit Research Paper No. 08-038.

Phelps, E. (1961). The golden rule of accumulation: a fable for growthmen. *The American Economic Review 51*(4), 638–643.

Ramsey, F. (1928). A mathematical theory of saving. *Economic Journal 38*(152), 543—559.

Romer, P. (1986). Increasing returns and long-run growth. *The Journal of Political Economy 94*(5), 1002.

Solow, R. M. (1956). A contribution to the theory of economic growth. *Quarterly Journal of Economics 70*(1), 56–94.

Topkis, D. M. (1998). *Supermodularity and Complementarity*. Princeton University Press.