Appendix

Contents

1 Methods in Detail

1.1 Creating the List of App Companies

As mentioned in our methodology section, the process of snowball-sampling companies within the Danish app development eco-system, originated from a list of 33 companies identified through prior ethnographic research. These companies served as our first-tier companies and provided the foundation for a snowball sampling strategy aimed at discovering additional relevant companies.

Using GitHub's advanced search, we queried each company to identify affiliated user profiles. To target users associated with the companies, we constructed search queries that combined a *keyword* derived from the company name (for example, omitting "ApS" from "Makeable ApS").

For companies with common or international names – such as PWC – initial searches returned thousands of results, only a fraction of which could be confidently linked to the company. To improve precision, we applied a location filter using GitHub's query qualifiers[1], incorporating keywords for the 10 biggest Danish cities, and variants of "Denmark," "Copenhagen". For instance, our query for *Netcompany* looked as the following:

```
"netcompany location:dk location:Denmark location:Danmark location:CPH location:KBH
location:Copenhagen location:Cop location:København location:Odense location:Aarhus
 location:Århus location:Aalborg location:Ålborg location:Esbjerg location:Randers
location:Kolding location:Horsens location:Vejle location:Roskilde location:Herning"
```

In a few cases – such as *FrontIt*, *Greener Pastures*, and *Adform* – no users were returned when location filters were applied. In these instances, we removed the filters to identify at least one affiliated user per company (which we manually verified). Although this reduced geographic certainty, it was deemed necessary for initiating our snowball sampling process. Finally, profiles retrieved through the 33 constructed GitHub query were classified as first-tier users.

From the first-tier users, we identified second-tier users by examining their GitHub activity and connections. Given that the second-tier user set contained thousands of accounts, we created a subset for review by filtering for both the presence of Danish geographic markers in the user's

[1] See: `https://docs.github.com/en/search-github/searching-on-github/searching-users`

biographical data and an entry in the GitHub location field. We then manually reviewed the biographical data of these second-tier users to identify mentions of additional companies. New companies were included in our final sample if they met two criteria: (1) they were Danish or had a Danish branch, and (2) they were involved in app development or related data infrastructure services.

This iterative process led to the identification of 31 additional companies. Each of these companies were queried in the same manner as the company on the initial list, producing further sets of first-tier and second-tier users to include in the final dataset.

1.2 Detailing the Scraping of Users from the GitHub REST API

For each user, we retrieved two categories of data from GitHub's REST API: (1) biographical information and (2) relational data based on user activity. Biographical fields included login, user type, company, email, location, bio, and blog. Except for location, which was used to infer geographic affiliation, the remaining free-text fields allowed us to identify connections to specific companies. All data was collected exclusively from public repositories and publicly visible profiles.

To reduce processing time and stay within API rate limits, we applied a threshold-filter based on the number of public repositories a user owned. We set this threshold at 300 repositories, informed by a review of company profiles from our initial list. Users exceeding this limit – typically high-activity, well-connected accounts outside the study's geographic scope – were skipped during scraping. All skipped users were logged for possible manual review.

Relational data represented ties between users through four types of activity: starring, watching, forking, and following. While following is directed at other users, the remaining actions target repositories. For each user, we collected both inbound (interactions received) and outbound (interactions initiated) connections. These ties formed the basis of the network used in our analysis.

Data collection was conducted using the `PyGithub package`[2], a Python wrapper for GitHub's REST API v3[3]. The API imposes rate limits based on authentication status: up to 30 search requests per minute[4] and 5,000 general requests per hour with an authenticated personal access token[5]. To comply with these limits, we implemented automated pauses to align with reset intervals.

1.3 Sorting for Danish App Company Profiles

For our final dataset, all user profiles were filtered based on two criteria: (1) geographic location in Denmark and (2) affiliation with a company from our sample. For first-tier users, only company affiliation was checked, as geographic location had already been addressed during the initial GitHub search. For second-tier users, they were intially filtered on only geographic location – this time requiring a more advanced location filter than for first-tier's (detailed below) – in order for new companies to appear. After the inclusion of new companies to our sample, the second-tier users are then filtered on affiliation as well.

The geographic filtering for second-tier users proceeded in two steps. First, a regular expression (regex) was applied to all biographical fields to identify matches with a predefined list of Danish city names (as used in the GitHub query example above). If no match was found, and the user had

[2] v1.58.0, `https://pygithub.readthedocs.io/en/stable/introduction.html`

[3] `https://docs.github.com/en/rest?apiVersion=2022-11-28`

[4] `https://docs.github.com/en/rest/search/search?apiVersion=2022-11-28`

[5] `https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28#about-primary-rate-limits`

an entry in the location field, this location string was evaluated for whether it corresponded to a place in Denmark using the GeoNames database[6]. Users matching in either step were categorized as based in Denmark, while those without a match were excluded from the sample.

To determine company affiliation, each company name was matched against the biographical data of a user profile – including the company, bio, and email fields – using a company-specific regex. These regex patterns were maintained in a dedicated script (`resources/regex_company_patterns.py`). In cases where a single user profile matched multiple company patterns, company affiliation was manually reviewed and assigned.

1.4 Creating the Networks

For each user in the dataset, inbound and outbound ties were recorded for four types of GitHub actions: forking, starring, watching, and following. This produced eight directional tie variables: `forks_in`, `forks_out`, `stars_in`, `stars_out`, `follows_in`, `follows_out`, `watches_in`, and `watches_out`. Each entry in these variables is a list of dictionaries, where each dictionary represents a connection and includes the creation time, the name of the targeted repository (if applicable), and the GitHub username of the counterpart involved in the interaction.

To construct the edgelists, we first created a unique list of Danish GitHub users affiliated with companies in our sample. We then looped over each user's activity data across the eight tie types. For each connection, a lookup was performed to check whether the other user was in the unique user list. If so, a directed tie was added to the edgelist, annotated with metadata including: company affiliations of both users, whether the tie was intra- or inter-company, the action type, repository name, and timestamp.

After removing duplicates, the dataset was split into two distinct edgelists: (1) A collaboration network, composed solely of fork ties, and (2) an attention network, composed of stars, watches, and follows.

Each edgelist was then transformed into a weighted user-to-user edgelist. For every directed pair of users, the number of observed ties in that direction was used as the weight. In the attention network, the three action types were combined into a single edge, but separate counts for each type (star, watch, follow) were stored as edge attributes.

Finally, both user-level networks were aggregated into company-level weighted edgelists. In these, edge weights represent the number of unique, directed user-to-user ties occurring between two companies. Counts of the underlying GitHub actions were also aggregated as edge attributes. Self-loops were allowed to capture intra-company ties and understand internal interaction patterns.

These two company-level edgelists form the basis for the directed network analyses presented in the main analysis section.

1.5 Plotting the networks and calculating network metrics

To visualize the collaboration and attention networks, we used the Python library `matplotlib`[7], the standard plotting tool in conjunction with the `NetworkX`[8] network analysis package.

In the plotted networks, nodes are colored according to the four company categories as described in the paper: (1) Digital and marketing consultancies, (2) Bespoke app companies, (3) Data brokers

[6] https://www.geonames.org/
[7] `https://matplotlib.org/`
[8] `https://networkx.org/`

and infrastructure providers, and (4) Companies with app-related services/products as part of a broader offering.

The added edge thickness was set to be proportional to edge weight: higher weights correspond to thicker lines. Self-loops are included to represent intra-company interactions.

For layout generation, we used the Fruchterman–Reingold algorithm, a force-directed layout algorithm implemented in NetworkX. This algorithm simulates a physical system where nodes repel each other like particles, while edges act as springs pulling connected nodes together. The simulation iterates until a stable equilibrium is reached, resulting in a layout that is typically more interpretable than deterministic or grid-based methods.

Two key parameters controls the layout: **k**, which determines repulsion strength between nodes. A higher value produces a more spread-out graph. We set k = 1.5 for the attention network and k = 1.1 for the collaboration network. Next, number of **iterations** controls the number of iterations used to compute equilibrium. We set this to 50, which was sufficient given the relatively small size of the networks.

2 Additional Descriptive Tables

2.1 Collaboration Network

Table 3: Count of intra- and inter-company collaboration connections (forking) among employees per company type (total count: 89)

		Target			
		1. Digital and marketing consultancies	2. Bespoke app companies	3. Data broker- and infrastructure companies	4. Companies with specific digital part/app as part of service/product
Source	1. Digital and marketing consultancies	78	0	0	2
	2. Bespoke app companies	3	3	0	0
	3. Data-broker- and infrastructure companies	0	0	3	0
	4. Companies with specific digital part/app as part of service/product	3	0	2	14

Table 4: Count of employee inter- and intra-connections per company in the collaboration network

Company	**Category**[ε]	**Total Inter Actions**	**Inter Inbound**	**Inter Outbound**	**Intra Forks**[b]
abtion	1	0	0	0	5
cbrain	4	4	0	4	2
charlie tango	1	0	0	0	1
delegateas	1	1	1	0	4
deon digital	3	2	2	0	0
ffw	1	0	0	0	1
house of code	2	1	0	1	0
jobindex	4	1	1	0	1
kmd	1	0	0	0	1
knowit	1	0	0	0	1
miracle	2	0	0	0	3
monstarlab	1	6	3	3	1
mustache	2	2	0	2	0
netcompany	1	2	1	1	8
nodes	1	14	11	3	13
nuuday	4	3	1	2	2
shape	1	1	1	0	13
siteimprove	3	0	0	0	3
skat	4	1	1	0	1
trifork	1	2	0	2	16
tv2	4	0	0	0	2
uptime	1	4	0	4	3
yousee	4	2	2	0	0
ørsted	4	4	1	3	2

[a] **Company Category:** 1 = Digital and marketing consultancies, 2 = Bespoke app companies, 3 = Data-broker- and infrastructure companies, 4 = Companies with specific digital part/app as part of service/product.
[b] As the inbound and outbound count for intra-connections on a company-level is the same, there is no need to differentiate. For the same reason no total count is displayed for the intra category.

2.2 Attention Network

Table 5: Count of intra- and inter-company attention connections (starring, watching, following) among employees per company type (total count: 601)

		Target			
		1. Digital and marketing consultancies	2. Bespoke app companies	3. Data broker- and infrastructure companies	4. Companies with specific digital part/app as part of service/product
Source	1. Digital and marketing consultancies	431	3	3	25
	2. Bespoke app companies	2	59	0	1
	3. Data-broker- and infrastructure companies	5	0	67	5
	4. Companies with specific digital part/app as part of service/product	20	4	12	172

Table 6: Count of employee inter- and intra-Connections per company in the attention network

		Inter									Intra[b]			
					Inbound			*Outbound*						
Company	**Category[a]**	**Total Actions**	**Total Inbound**	**Total Outbound**	**Inter Stars**	**Inter Watches**	**Inter Follows**	**Inter Stars**	**Inter Watches**	**Inter Follows**	**Total Actions**	**Intra Stars**	**Intra Watches**	**Intra Follows**
abtion	1	2	1	1	0	1	0	0	1	0	21	5	15	1
capgemini	1	0	0	0	0	0	0	0	0	0	1	0	0	1
cbrain	4	18	8	10	4	1	3	5	2	3	5	2	3	0
charlie tango	1	2	2	0	2	0	0	0	0	0	18	5	8	5
codefort	1	1	0	1	0	0	0	1	0	0	0	0	0	0
commentor	2	3	2	1	1	0	1	0	0	1	1	0	1	0
creuna	1	4	2	2	1	1	0	1	1	0	5	2	3	0
delegateas	1	1	1	0	1	0	0	0	0	0	17	5	8	4
deon digital	3	19	12	7	8	2	2	5	2	0	28	9	14	5
dgi-it	4	1	0	1	0	0	0	0	1	0	2	1	1	0
eg a/s	1	0	0	0	0	0	0	0	0	0	6	0	1	5
ffw	1	0	0	0	0	0	0	0	0	0	5	1	4	0
holion	1	0	0	0	0	0	0	0	0	0	2	0	2	0
house of code	2	4	4	0	1	3	0	0	0	0	1	0	1	0
immeo	1	0	0	0	0	0	0	0	0	0	2	0	2	0
jobindex	4	32	13	19	5	4	4	8	4	7	22	5	8	9
kmd	1	2	1	1	0	1	0	0	0	1	8	0	6	2
knowit	1	4	2	2	1	1	0	0	2	0	2	0	2	0
kruso	1	1	0	1	0	0	0	0	1	0	2	1	1	0
makeable	2	1	1	0	1	0	0	0	0	0	1	0	1	0
miracle	2	2	0	2	0	0	0	1	1	0	50	1	48	1
monstarlab	1	19	7	12	2	3	2	5	5	2	1	0	1	0
mustache	2	0	0	0	0	0	0	0	0	0	4	1	3	0
nabto	3	0	0	0	0	0	0	0	0	0	2	0	2	0
netcompany	1	28	4	24	1	2	1	7	9	8	49	8	29	12
nodes	1	36	23	13	10	9	4	4	3	6	34	16	14	4
nuuday	4	21	9	12	4	3	2	5	5	2	27	6	12	9
oxygen	1	7	4	3	0	2	2	0	2	1	4	1	3	0
pentia	1	4	3	1	0	3	0	0	1	0	18	2	15	1

Table 6: (continued)

		Inter									Intra[b]			
					Inbound			*Outbound*						
Company	**Category[a]**	**Total Actions**	**Total Inbound**	**Total Outbound**	**Inter Stars**	**Inter Watches**	**Inter Follows**	**Inter Stars**	**Inter Watches**	**Inter Follows**	**Total Actions**	**Intra Stars**	**Intra Watches**	**Intra Follows**
reepay	3	0	0	0	0	0	0	0	0	0	2	0	2	0
relatel	4	2	0	2	0	0	0	1	0	1	7	0	4	3
saxo bank	4	1	0	1	0	0	0	1	0	0	2	0	2	0
shape	1	11	8	3	4	1	3	0	2	1	70	17	43	10
signifly	1	2	2	0	1	0	1	0	0	0	7	1	5	1
siteimprove	3	6	3	3	0	3	0	1	2	0	35	5	17	13
skat	4	18	17	1	2	6	9	0	1	0	2	0	2	0
snapp	2	0	0	0	0	0	0	0	0	0	2	0	2	0
strømlin	1	1	1	0	0	1	0	0	0	0	0	0	0	0
studiesandn	4	0	0	0	0	0	0	0	0	0	1	0	0	1
systematic	1	1	1	0	0	1	0	0	0	0	5	1	4	0
trifork	1	3	2	1	2	0	0	1	0	0	93	24	64	5
tv2	4	12	1	11	0	0	1	3	6	2	54	15	18	21
uni-soft	4	0	0	0	0	0	0	0	0	0	2	0	0	2
uptime	1	5	1	4	1	0	0	3	0	1	23	6	5	12
yousee	4	8	7	1	1	6	0	0	0	1	8	3	4	1
ørsted	4	16	7	9	2	0	5	3	3	3	9	2	4	3

[a] **Company Category:** 1 = Digital and marketing consultancies, 2 = Bespoke app companies, 3 = Data-broker- and infrastructure companies, 4 = Companies with specific digital part/app as part of service/product.
[b] As the inbound and outbound count for intra-connections on a company-level is the same, there is no need to differentiate.