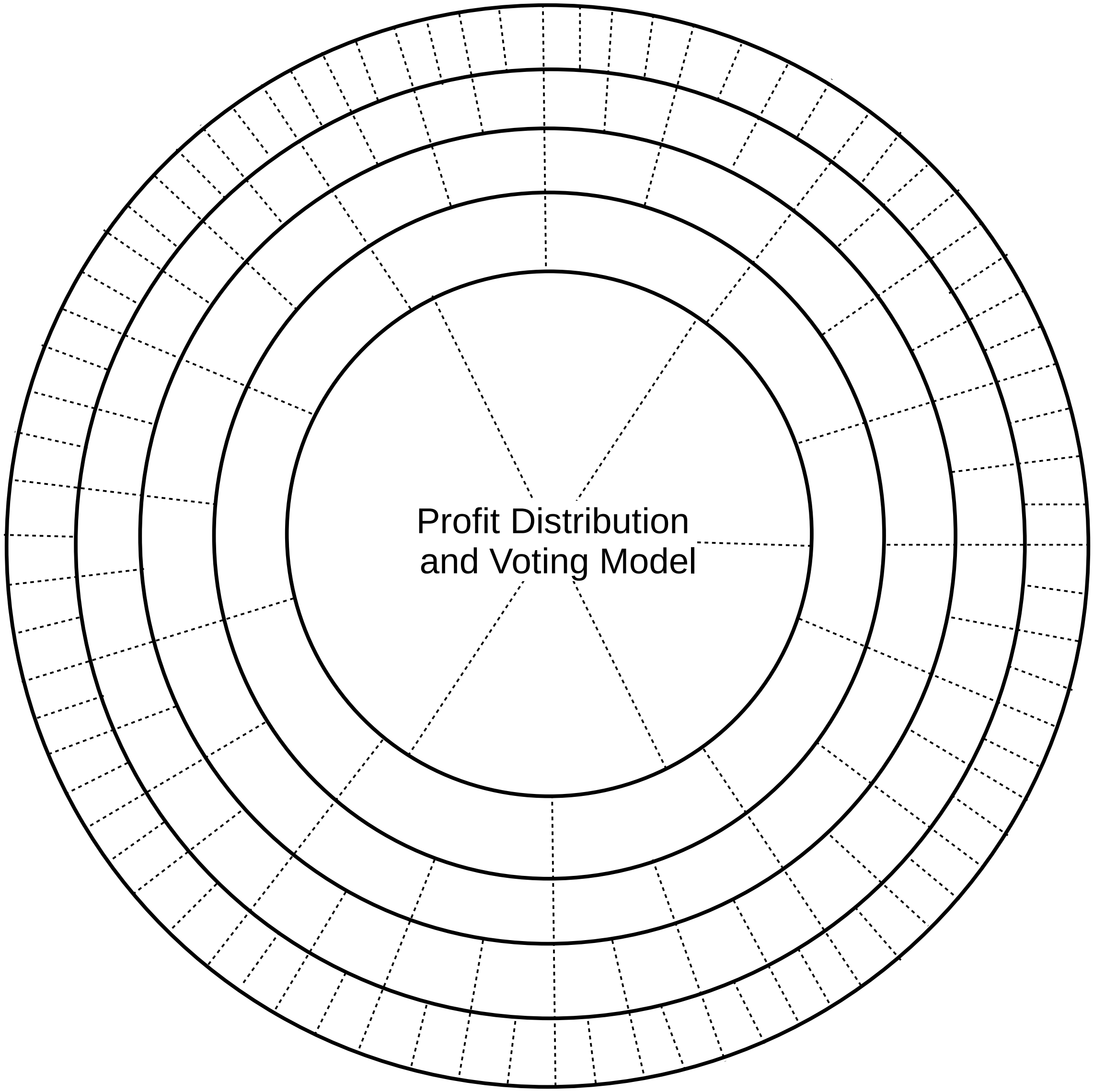
Innovation Corporation



Innovation Corporation

There are two components, a **voting model** and a **profit distribution model**.

The company owns itself; shares are only allotted to positions within the company and are only for voting rights.

Equity capital could be raised in subsidiary entities so long as all employees are part of the parent I corporation.

The purpose of the I corporation model is to have a company where those who are putting in the work and expertise are the ones deciding their future. Wealth is ethically, incentively distributed within the company; the hope is to attract the smartest minds, and their innovation will lead the way.

Justin Mason

Voting Model
Corporate Structure

Goal: Those within the company are the ones making decisions and electing their leadership.

-Shares are not owned; they are assigned to employee positions within the company. They are exclusively for voting rights and not profit distribution.

-Voting is done confidentially, preferably on block chain. Shareholders are obligated to vote, and a yearly vote is taken on whether to hold an election.

-The chairman must appoint the first 4 other members of the board of directors. After that the directors (ring 0), including the chairman, are elected by shareholders in a ranked choice vote.

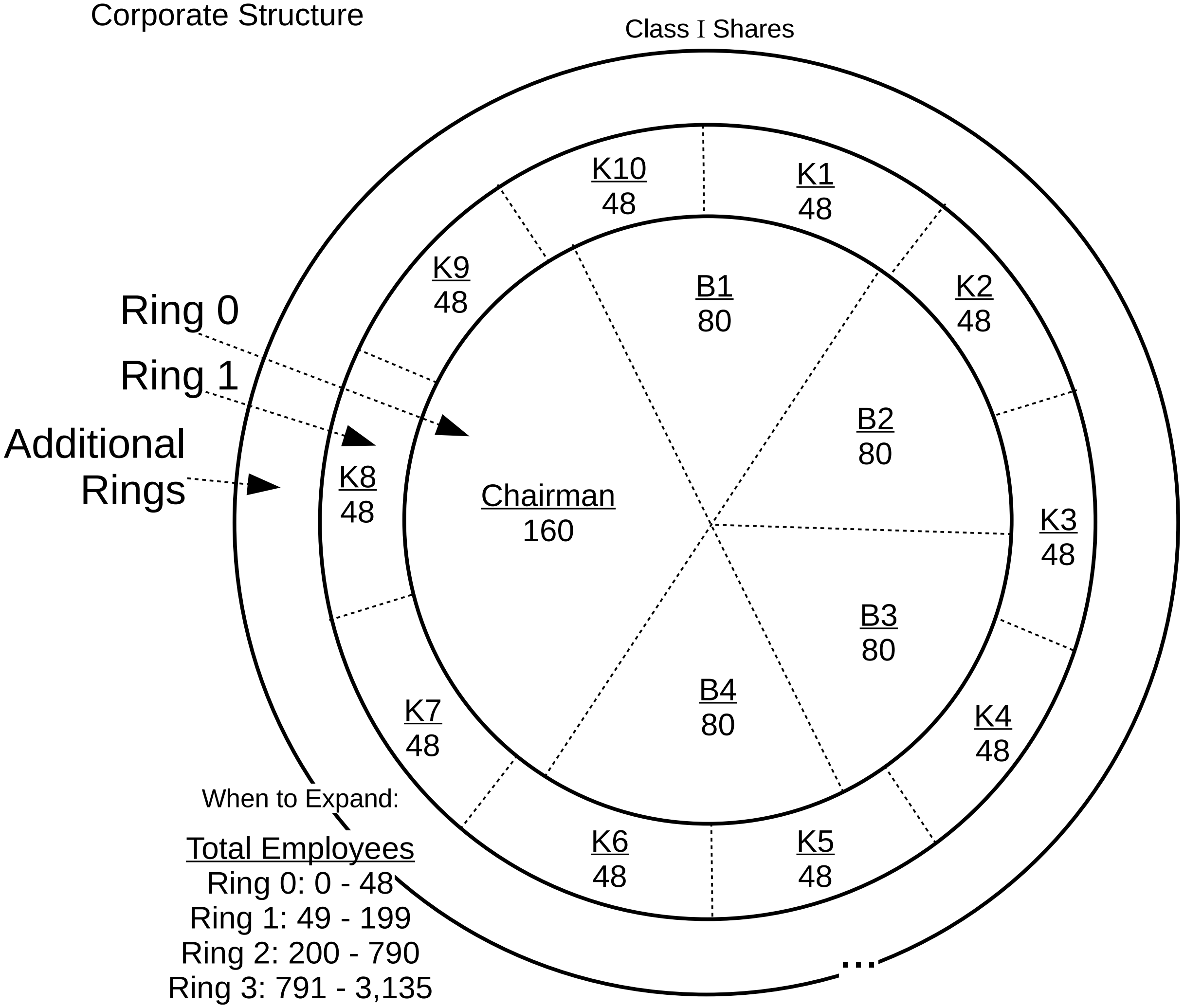
-Key positions (rings 1-4) are appointed by the board of directors. Corporate officers are prioritized if they are not also on the board of directors.

-50/50 ties are broken by the chairman.

-Each ring has 480 shares. With the exception of Ring 0, they are split evenly among the slots.

-Each ring acts as an expansion to the companies number of key positions in relation to total employees. Ring expansion can occur sooner, but not later than as is defined in the table.

-New rings must be filled within 1 year, vacated slots within current rings must be filled within 1 month. All key positions must be filled for a vote, and all board of director nominations must be existing shareholders.



When to Expand:

Total Employees
Ring 0: 0 - 48
Ring 1: 49 - 199
Ring 2: 200 - 790
Ring 3: 791 - 3,135
Ring 4: 3,136 - 12,475

Slots/Shares

Ring 0: 5 slots / see above
Ring 1: 10 slots / 48 shares
Ring 2: 20 slots / 24 shares
Ring 3: 40 slots / 12 shares
Ring 4: 80 slots / 6 shares

All slots do not exceed Dunbar’s Number.
(Ring 4)

Profit Distribution Model

Justin Mason

Goal: Share profits while incentivizing key leadership positions and innovation.

-The voting model results in a disparity of 26x between the chairman and the outermost ring; this also serves as the pay ratio limit in the profit distribution model (This is similar to the U.S. in the 60s and 70s). Some exclusions might apply for internships or temporary employees, but this is not to be abused for the sake of profit.

-The chairman's total compensation plus profit distribution cannot exceed 26x the lowest paid employee's compensation/profit.

-Steps 0-4 align with rings 0-4 (if they exist).

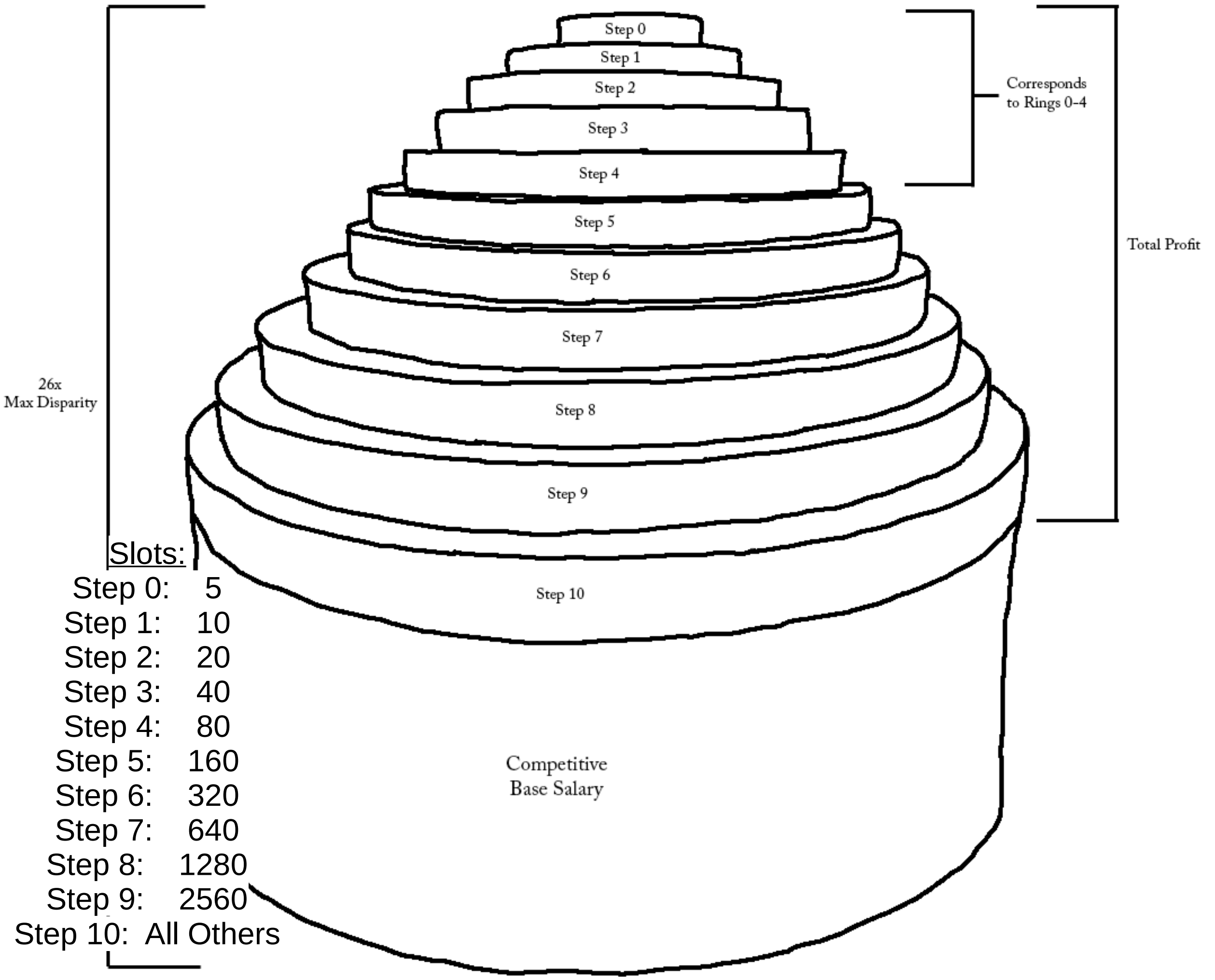
-Each step up provides a greater incentive (portion of the profit) to advance within the company. While steps 0-4 correspond to the rings, the other steps are left to boards discretion to incentivize certain job positions.

-All employees are slotted into a step. Their base salary is industry competitive for whatever job position they are filling. The profit distribution is treated strictly as a bonus.

-If an employee occupies a slot for a portion of the year, they get that portion of the profit distribution.

-If the company is ever dissolved, the assets are converted to cash and distributed as profit.

-Solving the model is described on the next page followed by the prototype code.



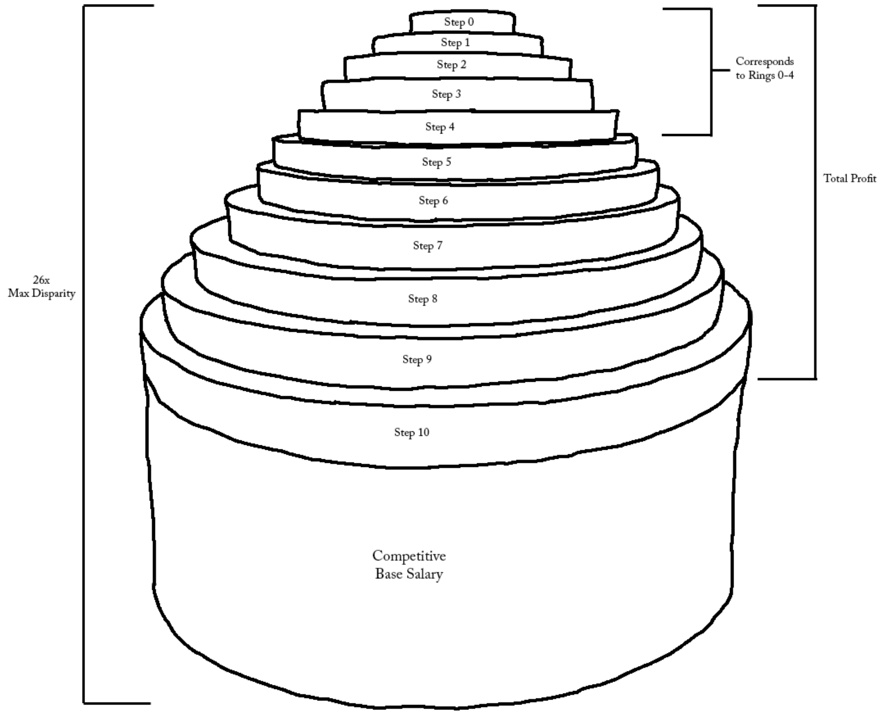
Slots:

Step 0:	5
Step 1:	10
Step 2:	20
Step 3:	40
Step 4:	80
Step 5:	160
Step 6:	320
Step 7:	640
Step 8:	1280
Step 9:	2560
Step 10:	All Others

number of steps = the amount needed to slot all employees

Profit Distribution Problem

Justin Mason



5*2^step

	Slots:	Total Employees:
Step 0:	5	5
Step 1:	10	15
Step 2:	20	35
Step 3:	40	75
Step 4:	80	155
Step 5:	160	315
Step 6:	320	635
Step 7:	640	1275
Step 8:	1280	2555
Step 9:	2560	5115
Step 10:	All Others	All Others

Steps are created until they hold all employees up to a max of 12,475.

Given the:
Total Profit
Total Employees
Lowest Employee Compensation
Highest Employee Compensation

Find a model that distributes the profit so that each employee gets proportionally more profit than an employee of a lower step with the following constraint:

(highest compensation + step 0) / (lowest compensation + lowest step) <= 26x

The pay disparity should approach 26x when the company is very profitable.

Version 1 Code
(prototype)

```
public static void calculateProfit(double totalProfit, double lowestCompensation, double highestCompensation, int steps, int
totalEmployees){
        int[] employeesPerStep = {0,0,0,0,0,0,0,0,0,0,0};
        int stepSum = 0;
        int stepSumDiff = 0;
        double baseMultiplier = 1;
        double currentMultiplier = 1;
        double[] profitMultiplier = {0,0,0,0,0,0,0,0,0,0,0};
        double profitDivisor = 0;
        double profitFactor = 0;
        double[] profitPerEmployee = {0,0,0,0,0,0,0,0,0,0,0};
        double profitRatio = 0;
        double payRatio = 0;
        //Scale profitRatioTarget to number of steps using (1,1) and (11,26)
        double profitRatioTarget = ((5.0/2.0)*(steps+1)-(3.0/2.0));
        //Fill in employees per step maximum
        for(int count = 0; count <= steps; count++) {
            employeesPerStep[count] = (int)(5*Math.pow(2,count));
            }
        //Find the total capacity of the steps
        for(int count = 0; count <= steps; count++) {
            stepSum += employeesPerStep[count];
            }
        //Find the difference from total capacity and number of employees, correct last step
        stepSumDiff = totalEmployees - stepSum;
        employeesPerStep[steps] += stepSumDiff;
        for(int count=0; count <= steps; count++) {
            System.out.print("\nStep " + count + " employees: " + employeesPerStep[count]);
            }
        System.out.print("\n");
        //Profit Multiplier
        for(;profitRatio < profitRatioTarget; baseMultiplier += .005) {
            //Calculate Profit Multipliers (Must reset currentMultiplier because of *=)
            currentMultiplier = 1;
            for(int count = steps; count >= 0; count--) {
                profitMultiplier[count] = currentMultiplier;
                currentMultiplier *= baseMultiplier;
                }
            //Calculate profitFactor (Must reset profitDivisor because of +=)
            profitDivisor = 0;
            for(int count = 0; count <= steps; count++) {
                profitDivisor += (employeesPerStep[count] * profitMultiplier[count]);
                }
            profitFactor = totalProfit / profitDivisor;
            //Profit Per Employee Per Step
            for(int count = 0; count <= steps; count++) {
                profitPerEmployee[count] = (profitMultiplier[count]*profitFactor);
                }
            profitRatio = (profitPerEmployee[0]/profitPerEmployee[steps]);
            }
```

```
//Subtract .005 to correct for final loop action and another .005 to reduce below 26x and Recalculate
        baseMultiplier -= .01;
        //Calculate Profit Multipliers (Must reset currentMultiplier because of *=)
        currentMultiplier = 1;
        for(int count = steps; count >= 0; count--) {
            profitMultiplier[count] = currentMultiplier;
            currentMultiplier *= baseMultiplier;
            }
        //Calculate profitFactor (Must reset profitDivisor because of +=)
        profitDivisor = 0;
        for(int count = 0; count <= steps; count++) {
            profitDivisor += (employeesPerStep[count] * profitMultiplier[count]);
            }
        profitFactor = totalProfit / profitDivisor;
        //Profit Per Employee Per Step
        for(int count = 0; count <= steps; count++) {
            profitPerEmployee[count] = (profitMultiplier[count]*profitFactor);
            }
        profitRatio = (profitPerEmployee[0]/profitPerEmployee[steps]);
        //Total Pay Ratio
        payRatio = ((profitPerEmployee[0] + highestCompensation)/(profitPerEmployee[steps] + lowestCompensation));
        //Display
        for(int count = 0; count <= steps; count++) {
            System.out.printf("\nStep %-2d Employee Profit: $%,.2f", count, profitPerEmployee[count]);
            }
        System.out.printf("\n\nProfit Ratio: %.2f" , profitRatio);
        System.out.printf("\n\nPay Ratio: %.2f" , payRatio);
        if(payRatio > 26) {System.out.print(" - !PAY RATIO EXCEEDED!");}
        System.out.print("\n\nCalculation Complete");
    }
```

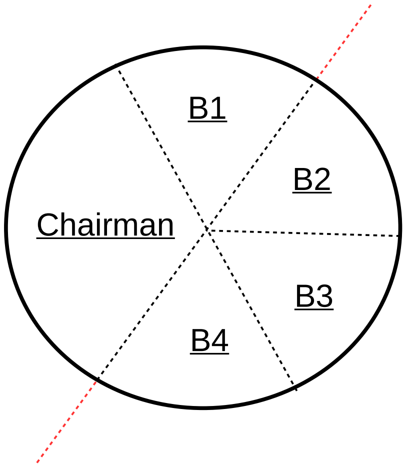
highest step = step 0

First the program calculates how many employees are going to be in each step.
The program assigns the lowest step a profit multiplier of one, and each successive step's profit multiplier is the previous multiplier times the base multiplier.
Then we multiply the number of employees in each step with their profit multiplier, and total up the results as the profit divisor.
The profit factor is the total profit divided by the profit divisor, and we multiply the profit factor by the profit multiplier of a step to determine how much profit each employee in that step gets.
We compare the individual profit of the highest step and lowest step to determine the profit ratio, and keep incrementing the base multiplier by .005 until the profit ratio approaches a scaled ratio of 26x.

Then, when we add the profit distribution to the highest paid and lowest paid employee's total compensation, we find the actual pay disparity which should scale fairly only approaching a 26x pay ratio in a large, profitable company.

Mathematical Significance

Everything is based on the initial, subjective premise that it is fair for the Chairman/Founder to have 50% of the votes if there is one board member that sides with them. The next ring dilutes this power, giving the board of directors 50% of the total votes instead and so on.



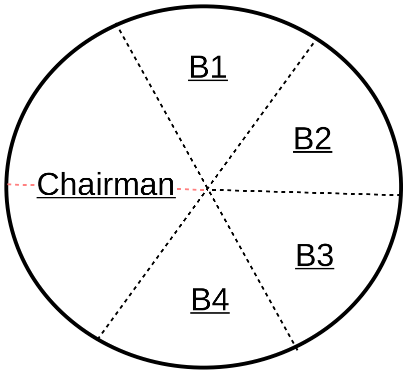
The second assertion is that the total number of decision makers should not exceed Dunbar’s Number. That number is the suggested cognitive limit of how many individual relationships a person can maintain. Each ring simply doubles the number of people, so:

Ring 0 = 5
Ring 1 = 10
Ring 2 = 20
Ring 3 = 40
Ring 4 = 80

Add those up and you get 155, any further expansion would exceed Dunbar’s Number.

Mathematical Significance

So now we know there will be 5 rings, and the outermost ring will have 80 people.



We figure out the number of shares by multiplying the factors 80 (outer ring) and 6 (inner ring).

So, 480 shares per ring.

The final question is when ring expansion should occur. At its maximum, this model would have 80 people at the outer ring. If these are the top tier of management, we’ll use Dunbar’s Number once again to say that each position can have 154 people below them. This gives us the fraction 155/80. We can use this to scale expansion proportionally.

R = Number of people in current outer ring.
$R^2*(155/80)$ = Number of employees in addition to those within the rings.

Total Employees
Ring 0: 0 - 48
Ring 1: 49 - 199
Ring 2: 200 - 790
Ring 3: 791 - 3,135
Ring 4: 3,136 - 12,475

I believe with Dunbar’s number, and two degrees of separation from the chairman, you can have a maximum of 12,475 people in a company that can still value community.

All companies wishing to claim status as an I-Corp must follow the guidelines provided in this document.

A 2/3 supermajority of shareholders can vote to adopt an updated I Corporation version, but there must be a majority vote of *all* employees to fundamentally restructure.