**Hiểu thế nào cho đúng về “liên chức năng”**

Nhóm Scrum có hai đặc điểm cực kì quan trọng là: tự tổ chức (self-organizing) và liên chức năng (cross-functional). Về tự tổ chức, tự quản, tự định hướng thì dễ hiểu, nhưng thế nào là “liên chức năng”? Bài này mong muốn chỉ ra  một số hiểu lầm thường gặp, cũng như đào sâu thêm về khái niệm này trong phạm vi Scrum Framework để người học và thực hành Scrum đỡ bị rơi vào các nghịch lí khi cố gắng đối lập Scrum với những gì hiện có.

Có người bảo *“cá nhân trong nhóm liên chức năng biết làm tất cả mọi việc, trong nhóm Scrum không có ai là tester hết, không cần designer trong nhóm Scrum nữa, chỉ có developer thôi”*.

Có người lại bảo *“nhóm liên chức năng có nghĩa là ai cũng có thể làm được việc thay thế người khác, không cần phân vai rõ ràng, một người có vấn đề sẽ không ảnh hưởng đến cả nhóm”*.

Có người thì cao hứng phát biểu *“một nhóm mà liên chức năng thì siêu năng suất”*.

Những nhận định trên trên đều không ổn, và chúng là các điển hình về các ngộ nhận khi đề cập đến nhóm Scrum.

**Vậy nhóm liên chức năng là gì?**

Đây không phải là khái niệm mới mẻ gì, và về cơ bản nó chưa có nhiều thay đổi trong cách định nghĩa: *đó là một nhóm bao gồm nhiều cá nhân với các chuyên môn khác nhau được kết hợp lại cùng làm việc hướng tới một mục tiêu chung*.

Trong đội dự án, các cá nhân có thể đến từ nhiều phòng ban chức năng khác nhau, cũng có thể xuất phát từ bên ngoài (khách hàng, các cá nhân có liên quan, chuyên gia tư vấn v.v.). Nhưng khi đã thành một đội (team), thì các cá nhân làm việc tập trung cho đội như là*một đơn vị (unit)* để hoàn tất mục tiêu chung.  Bên trong nhóm liên chức năng không có các nhóm nhỏ khác. Ví dụ: một Nhóm Scrum “Alpha” được thành lập với một Product Owner, một Scrum Master, 2 Tester, 5 Developer, 1 Architect, 1 UX Designer sẽ không phân chia chức năng thành các nhóm nhỏ khác như nhóm Testing (2 người), nhóm Developement (5 người) … nữa, mà chỉ có một nhóm duy nhất “Alpha” với các cá nhân có các chuyên môn khác nhau, hợp thành một đội thống nhất để làm việc hướng tới sản phẩm cần phát triển. Trong cách nói của Ken và Jeff, tester hay analyst … đều là developer, họ là nhà phát triển nhưng có chuyên môn đặc thù là kiểm thử hay phân tích; developer trong trường hợp này không có ý nghĩa là coder\programer. Việc xóa nhòa các title này có mục đích là để gom mọi người vào một mục tiêu chung, không phân biệt “nhãn mác” (title): phát triển phần mềm.

Khác với nhóm liên chức năng, nhóm chức năng (functional team) thường chỉ phụ trách một loại công việc đặc thù. Ví dụ, phòng thiết kế thì không code, phòng test thì không có ai design. Công việc của nhóm chức năng thường có tính cô lập cao.

**Có phải cứ “liên chức năng” thì nhóm sẽ năng suất hay không?**

Hỏi cách khác, liệu cứ thành lập một nhóm Scrum với cách tổ chức liên chức năng như thế, thì nhóm sẽ ngay tức khắc sẽ gia tăng năng suất lao động?

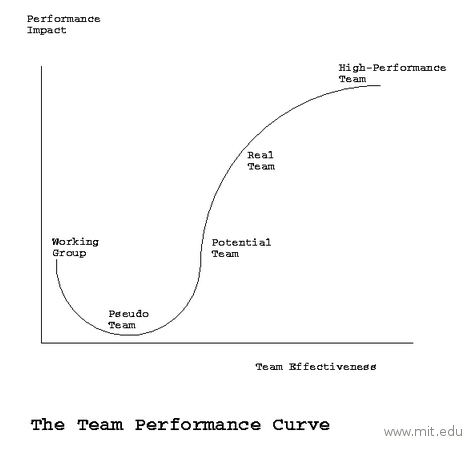
Rõ là không dễ dàng như thế rồi.

Liên chức năng chỉ là một “cấu hình” trong các cách thức cấu tạo nhóm làm việc. Và cách cấu hình này cho phép hình thành một đội độc lập với đầy đủ các chuyên môn cần thiết để hoàn tất công việc, dưới dạng một ‘business unit’ ở mức tối giản. Sự độc lập này tạo điều kiện cho nhóm ra các quyết định kịp thời, chính xác và nhanh chóng mà không bị phụ thuộc các đơn vị khác. Ví như trong cách thức tổ chức các nhóm chức năng (functional teams), một công việc phải đi qua nhiều công đoạn, mỗi công đoạn lại do các nhóm [với các chức năng] khác nhau đảm nhiệm, và vì thế sự phụ thuộc (dependency) gia tăng, khiến cho sự linh hoạt giảm xuống, và có nguy cơ làm giảm sự hiệu quả cũng như năng suất của nhóm.

Về mặt lí thuyết, với cấu hình “liên chức năng”, Nhóm Scrum có khả năng trở thành một “work cell” và có được “luồng một sản phẩm” (thuật ngữ của Lean), từ đó vừa gia tăng năng suất, vừa gia tăng chất lượng sản phẩm; đồng thời tạo điều kiện để vận hành “hệ thống kéo” (cũng thuật ngữ Lean), giúp loại bỏ các lãng phí không cần thiết, tối ưu hóa giá trị.

Nhưng một nhóm bất kì đều phải trải qua các giai đoạn Hình thành>Bão tố>Ổn định>Hiệu suất>Thoái trào (theo Tuckman). Có nghĩa là nó không thể đạt được hiệu quả ngay khi mới thành lập được.

Mặt khác, về “hiệu suất”, Katzenbach và Smith chỉ ra rằng một nhóm phải mất khá nhiều nỗ lực mới đạt được thành quả thực sự. Nó sẽ phải trải qua các trạng thái  như là một “nhúm” các cá nhân rời rạc, cho tới khi trở thành một nhóm thực sự, rồi mới phát huy hiệu quả cao.



*The team performance curve. Katzenbach and Smith (1993)*

Hiệu suất hay không do nhiều nguyên nhân cấu thành. Hiệu suất thường là cái cuối cùng trong tất cả các nỗ lực, mà “cấu hình nhóm” là một trong các nỗ lực như vậy. Nếu coi “cấu hình nhóm” là phần “cứng”, thì còn nhiều yếu tố nữa quyết định nhóm có “hiệu suất” không, chính là các phần “mềm” của nhóm: cách vận hành nhóm, quy tắc, phương pháp, sự lãnh đạo v.v. Trong Scrum, các yếu tố “mềm” này được phân bố rải rác trong chính sách trao quyền để nhóm “tự tổ chức” (self-organizing), “tự quản” (self-managing), “tự định hướng” (self-directed); trong các phân bố vai trò trong nhóm (ai lãnh đạo việc gì, ai phụ trách việc gì); các quy tắc và công cụ cộng tác (các burndown chart, backlogs, events, metrics); cho đến nguyên lí và cơ chế đảm bảo sự minh bạch trong làm việc v.v. Tất cả các yếu tố đó kết hợp nhuần nhuyễn với “cấu hình” mới tạo nên một nhóm cộng tác chặt chẽ, mau trưởng thành, sớm đạt được trạng thái năng suất cao. Lấy ví dụ, việc một nhóm liên chức năng được trao quyền tự tổ chức sẽ phát huy được sự chủ động, giảm thiểu phụ thuộc vào bên ngoài (hay bên trên); cũng do một nhóm liên chức năng đã có đủ các kĩ năng cần thiết để giải quyết vấn đề nên hoàn toàn có thể trao quyền cho nó để tự nó phát huy năng lực tập thể giải quyết vấn đề theo cách tốt nhất; nhờ đó sự liên chức năng kết hợp với sự tự quản, tự tổ chức có thể thúc đẩy các sáng kiến từ dưới lên (bottom-up), trực tiếp, nhanh chóng và chính xác. Khi đó, lãnh đạo\quản lí  thực hiện công việc chủ yếu là xúc tiến quá trình cộng tác nhóm và tự quản của nhóm thông qua đảm bảo quy trình, loại bỏ trở ngại, thiết lập và duy trì môi trường thuận lợi, thúc đẩy tiến trình thanh tra-thích nghi, v.v. chứ không còn bám sát từng đầu công việc nữa. Việc này đã được quy định chặt chẽ trong “mô tả công việc” của từng vai trò (Scrum Master, Product Owner, DevTeam) rồi.

Thế còn chuyện “liên chức năng thì không bị ảnh hưởng bởi xáo trộn cá nhân”? Chuyện một cá nhân ra đi mà nhóm không có xáo trộn chỉ là ảo tưởng. Chỉ có robot mới có thể đáp ứng được yêu cầu đó. Theo hướng đó thì chỉ việc đề cao quy trình, quản lí thật chặt đầu công việc của member theo lối “công nhân” tay chân, giao cho các công việc cố định, xếp vào các vị trí và công đoạn được mô tả rõ ràng  Việc này khả thi ở chỗ khác, chứ tuyệt nhiên khó hiệu quả ở đội phát triển phần mềm vốn có bản chất phức tạp, và rất đa dạng trong từng công việc. Hơn thế nữa, việc gán chuyện “không xáo trộn” như vậy cho nhóm liên chức năng còn phản lại nguyên lí của Agile: “Cá nhân và tương tác hơn là quy trình và công cụ”. Cá nhân là tiền đề của nhóm, là nơi quyết định thành bại của nhóm. Nhưng cá nhân ấy là cá nhân đặt trong nhóm cộng tác, với các đòi hỏi cao về các “tương tác” có chất lượng thì mới giải phóng được năng lực của chính các cá nhân ấy. Còn “quy trình” chỉ là cái hỗ trợ cho việc phát huy năng lực của cả nhóm. Nhóm làm việc bao giờ cũng hướng tới sự gắn kết cao độ, càng gắn kết thì lại càng dễ bị xáo trộn bởi sự thay đổi của cá nhân trong nhóm. Nhóm liên chức năng có thể có một ưu điểm trong việc hỗ trợ nhau trong công việc (do đòi hỏi cộng tác chặt chẽ, sự minh bạch trong công việc, cũng như đó là một nhóm-học-tập liên tục), nên có thể giảm thiểu rủi ro của việc mất đi “chuyên gia”, nhưng việc xáo trộn, thậm chí sốc, là không thể tránh khỏi.

Cuối cùng, có phải trong Nhóm Scrum không còn tester, không còn QA nữa, mà chỉ còn developer? developer phải là dân “xịn” biết test ngon, biết code giỏi, biết design khéo? Cũng là ảo tưởng nốt! Và ảo tưởng này xuất hiện đơn giản là xuất phát từ việc hiểu nhầm khái niệm “cross-functionality” như đã nói ở bên trên.