Expert-expert increasing each other's productivity:

(one person introduces the task)
i.e. Our boss wants to make the Fibonacci function.
(other person asks if the function should be recursive)
(one person comments on the inefficiency of a recursive fib function with large numbers)
(other person suggests that the recursive function could remember previous fib numbers)
(one person says that's is a great idea b/c it increases efficiency)

Expert-novice with the novice learning and sharing his creativity:

(expert introduces the task)
(Novice asks how to get started)
(expert explains how to use the divide and conquer principle)
(Novice verbally applies the principle)
(Expert compliments the novice on his progress)
(Expert also adds his analysis)
(Novice suggests how to improve the idea using his own creativity)

Novice-novice both learning

(One person introduces the topic)
(other person identifies parts that he is unsure about)
(One person explains to other person the parts he understands)
(One person also raises questions about parts he is unsure about)
(other person explains to the one person the parts he understands)
(Both compile a list of questions to ask their coach)

----------Version 2----------

Setting(to help the acting): Making a Fibonacci function that returns a string of sequence numbers k to n to be utilized in a graphing calculator simulator.

ex) if k = 3 & n = 5 then f(3, 5) = "2, 3, 5"

Fibonacci Sequence Definition: A sequence of numbers where the current number is the sum of the last 2 preceding numbers.

f(i) = f(i-1) + f(i-2) where i >= 2, else f(0) = 0, f(1) = 1.

Expert-Expert

E1: Ok so we need to make the Fibonacci function for this project now.
E2: Yes, I was thinking of doing the standard recursive approach.
E1: That sounds good, but recursion could be an issue if we have a lot of method calls and get a stack overflow.

E2: True, what if we made two functions one recursively and one with a loop for the larger calls.

E1: that sounds good and we can simultaneously test the methods results with each other to see if they function properly.

E2: I like that, let's start with the recursive method first though.

E1: ok

E1: <starts typing>

E2: I think the recursive methods base case should be tested after the standard case so it's not called after every method call.

E1: Yes that does sound like a proper way to go ill type it in now.

Expert-Novice

E1: Ok so we have to make a Fibonacci method for this project.

N2: A Fibonacci method is that a sequence thing?

E1: Yes it's a sequence of numbers where the current number is the sum of the last 2 preceding numbers.

N2: O ok so how can we make a method do that?

E1: Well I was thinking of using a recursive approach that will build up the sequence as the methods are called.

N2: That sounds better than anything I could of come up.

E1: <Starts Typing>

N2: <While reading method> But wait won't that calculate all the numbers up to n even when k is relatively close to n?

E1: Yes it will hmm I haven't thought of that, yes in larger cases of n and k this method would be highly inefficient.

N2: What if we could have some way to remember Fibonacci numbers so we wouldn't have to constantly re calculate them?

E1: Yes we could make a simple yet efficient database of spaced Fibonacci numbers to branch from with large cases of n, that's a great idea.

E1: <Types it in>

Novice-Novice

N1: What's a Fibonacci method again?

N2: Not sure let's look it up.

N2: <looks up function>

N2: Ah I see it's a sequence of numbers where the current number is the sum of the last 2 preceding numbers.

N1: So how are we going to make that.

N2: Well I've worked with loops before I bet we could make it with that

N1: Ok let's try that

N1: <Starts Typing>

N2: I'm not quite sure what's your coding there why do you think it needs two loops?

N1: Well I was thinking that sense a Fibonacci number is the sum of two other methods we would need two loops, one for each... right?

N2: I'm not sure about that lets ask the manager/teacher if that's even the right direction?

N1: ok